PLASTIC

IST STREP Project

SERVICE MODEL EDITOR USER GUIDE 1.1 (2008.03.13)

http://www.ist-plastic.org

Document History

Version	Type of change	Author(s)
v1	First version.	Luca Berardinelli (UDA)

1 INTRODUCTION 1-5

2 REQUIREMENTS 2-5

3 PLASTIC SERVICE MODEL : A COLLECTION OF VIEWS 3-8

3.1 MAGICDRAW DSL CUSTOMIZATION 3-9
3.1.1 CUSTOM DIAGRAMS AND CUSTOM TOOLBARS. 3-9
3.1.2 CUSTOMIZED CONTEXT MENUS AND CUSTOMIZED SPECIFICATION DIALOGS 3-10
3.2 PERFORMANCE, RELIABILITY AND MOBILITY REFERENCES 3-12
3.3 THE VACATION PLANNING SERVICE 3-12
3.3.1 THE VACATION PLANNING SERVICE SPECIFICATION 3-12
3.3.2 OPEN THE PLASTIC SERVICE TEMPLATE 3-12
3.3.3 VACATION PLANNING SERVICE MODEL 3-16
3.3.4 THE REQUIREMENT VIEW 3-17
3.3.5 THE SERVICE VIEW 3-21
3.3.6 THE COMPONENTS VIEW 3-26
3.3.7 THE IMPLEMENTATION VIEW 3-32
3.3.8 THE DEPLOYMENT VIEW 3-34

4 REFERENCES 4-39

List of Figures

Fig 1: MagicDraw required version....2-5
Fig 2: The taxonomy of PLASTIC custom diagrams 3-9
Fig 3: The customized toolbar of the Use Case Service Diagram....3-10
Fig 4: two examples of cutomized diagrams and their own customized toolbar: the Use Case Service Diagram and Service Description Diagram.....3-10
Fig 5: how to add quantitative additional information about reliability (REuser) or mobility (PHMobEntity)..3-11
Fig 6: customized specification dialog for an Actor with multiple stereotypes applied: <<ServiceConsumer>> , <<REuser>> (reliability) and PHMobEntity (mobility).....3-11
Fig 7: the overall structure of the UML PLASTIC Service Model.....3-14
Fig 8: how to select the PLASTIC Template.....3-14
Fig 9: the default package structure displayed within MagicDraw.....3-15
Fig 10: All the allowed views of the PLASTIC Service 3-15
Fig 11: The VPS Service Model overall structure. 3-16
Fig 12: The VPS Requirement View structure.....3-17
Fig 13: VPS Use Case Service Diagram.....3-18
Fig 14: Using the contextual menus to define the Requirement View. 3-18
Fig 15: Using the custom diagram and its customized toolbar to define the Requirement View. 3-19
Fig 16: The VPS Service View.....3-21
Fig 17: The VPS Service View::Structural View....3-22
Fig 18: The VPS Service Service Description Diagram. 3-23
Fig 19: Using the contextual menus or the customized toolbar to define the Service View. 3-23
Fig 20: The BPDD for the ConversationSpecification associated to the Hotel Reservation Use Case 3-24
Fig 21: The VPS Component View....3-26
Fig 22: The Service Specification Diagram for the Hotel Service.....3-27
Fig 23: The Service Specification Diagram for the Hotel Service.....3-28
Fig 24: The Component View::Behavioral View of the VPS....3-29
Fig 24: The Elementary Service Dynamics Diagram for the Hotel Service....3-30
Fig 25: The VPS Implementation View.....3-32
Fig 26: The VPS Component Design Diagram.....3-33
Fig 27: The VPS Deployment View....3-34
Fig 29:The Physical Mobility Pattern Diagrams of John Physical Mobility Entity.....3-36
Fig 30:The Service Description Diagrams at John's Office.....3-37

1 Introduction

MagicDraw® is a visual UML modeling and CASE tool: it is devised for the analysis and design of Object Oriented (OO) systems and databases. Within the PLASTIC project it has been adopted as Service Model Editor (SME) in order to specify the UML side of the PLASTIC Service Model.
The key features behind the choice of this UML tool are:

- The provision of a model driven approach (DSL Customization Engine®) for adapting domain specific profiles (i.e. PLASTIC Profile for adaptable context aware software service's domain) to create custom diagrams, custom specification dialogs, custom real-time semantic rules and other. The user is able to create a specialized domain-specific tool (Domain Specific Environment, DSE) and hide UML underneath.
- The seamless integration with Eclipse IDE and the open source Eclipse MDT project: MagicDraw provides import/export functionalities to support the free Eclipse UML file format (.uml). The exported Eclipse-based Service Model represents the input model that feeds model to (Analysis) model and model to (adaptable Java) code transformations.

Detailed information about MagicDraw® and its features is available on MagicDraw user manuals (<magicdraw installation dir>\manual) and on its official website (www.magicdraw.com).

2 Requirements

This section contains the requirements to install and set up the PLASTIC Service Model Editor

(i) **MagicDraw UML 14.0** (or higher) Standard Edition (or higher) with full Commercial License (or full Evaluation key) properly installed. It is worth to note that if you save your model with a newer version of MagicDraw you are not able to open it with any other older versions.

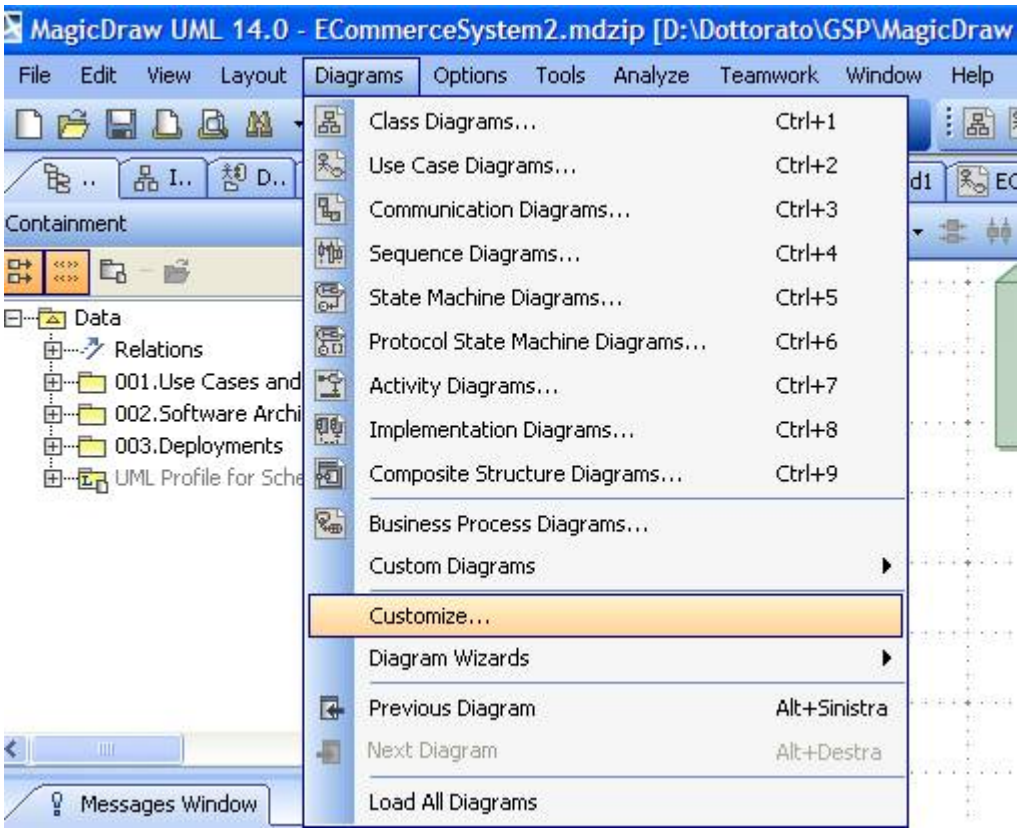
Fig 1: MagicDraw required version

(ii) The archive provided with this user manual:
   a. **Profiles.Plastic.rar**: it has to be uncompressed into <md install dir>/profiles
   b. **Samples.Plastic.rar**: it has to be uncompressed into <md install dir>/samples;
   c. **Templates.Plastic.rar**: it has to be uncompressed into <md install dir>/templates

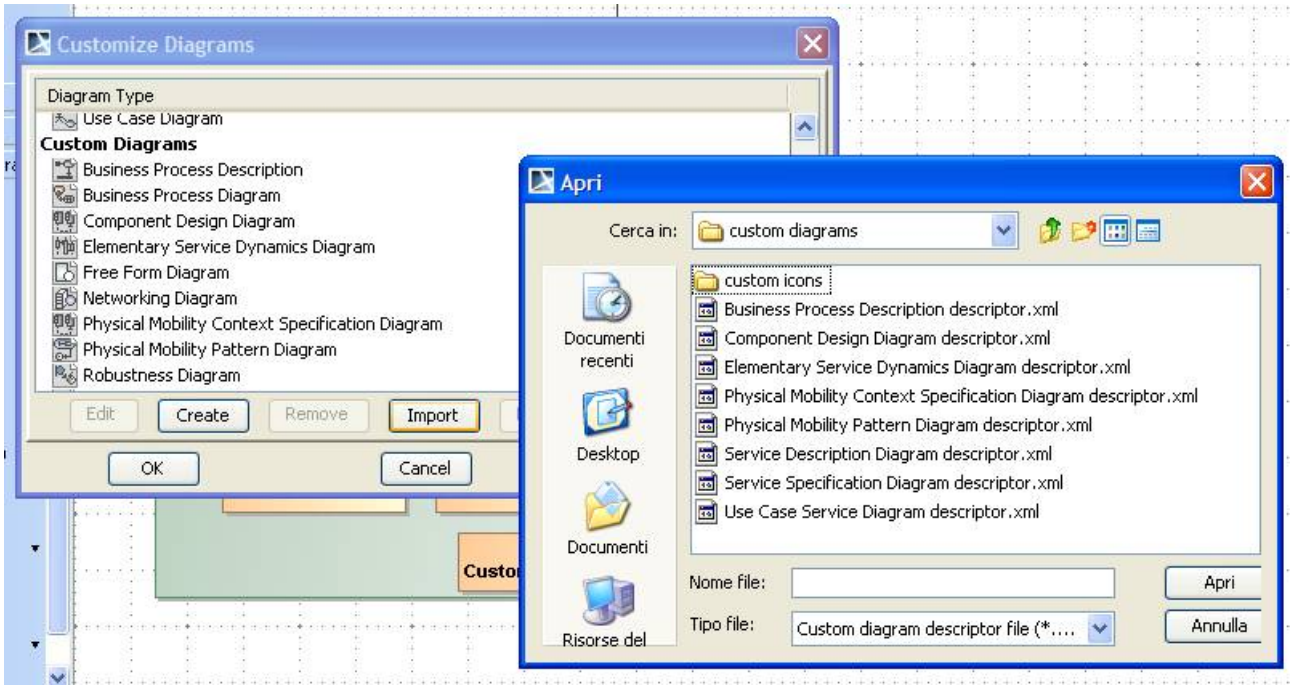
Then we have to import diagram descriptor files from <md install dir>/profiles/custom diagrams/.

1. Open MagicDraw;

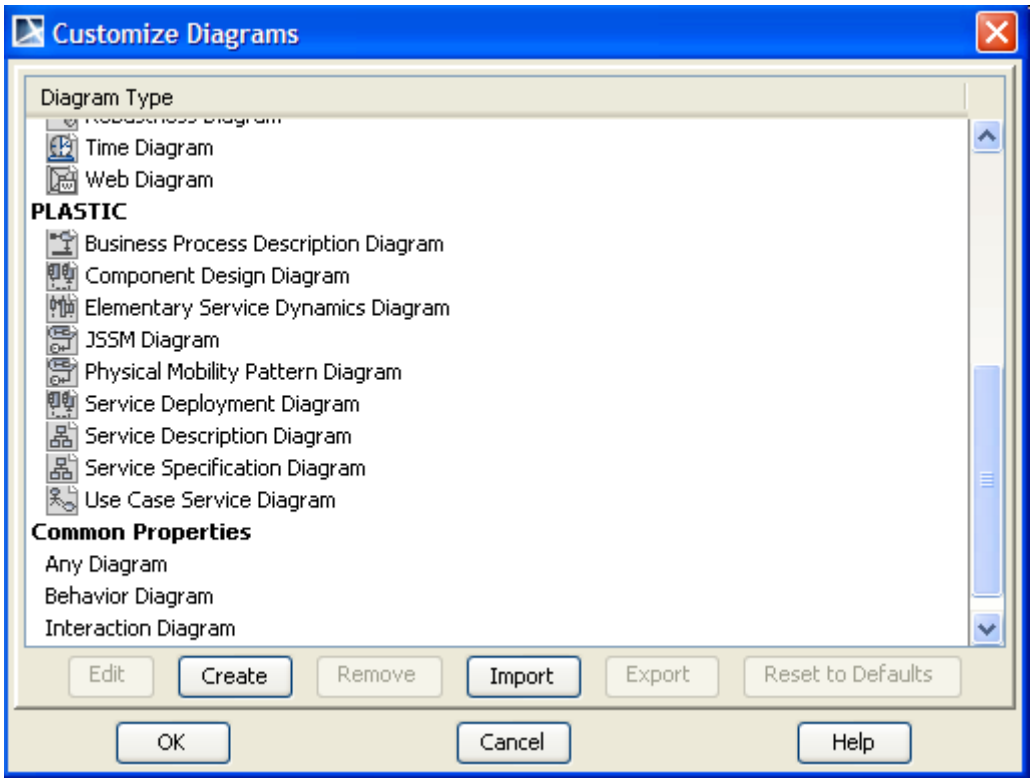
2. Click on Diagrams>Customize on the MagicDraw menu;



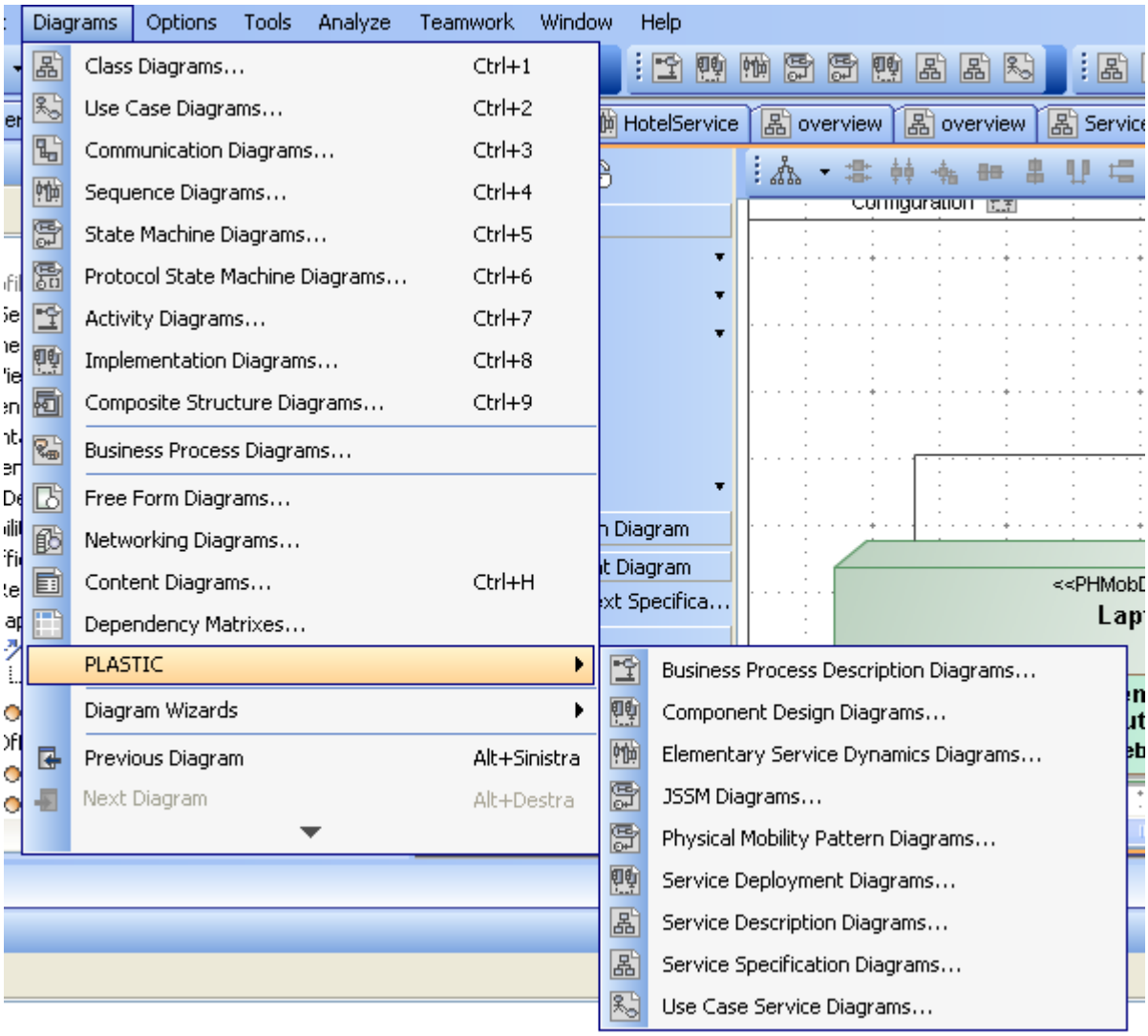
3. Click on the Import button the open the <magicdraw installdir>/profiles/PlasticProfile/custom diagrams;



4. Select, one a time, a descriptor file and push the Open button. You'll see the name of the plastic custom diagram just imported under the a new "PLASTIC" section within the "Customize Diagram" window.



5. MagicDraw® has to be **restarted** after the installation of these archives in order to activate the customization features (custom diagrams and custom toobars) provided by the PLASTIC Profile.



3 PLASTIC Service Model : a collection of views

The Plastic Service Model is organized in views (Fig 7). The following sections describe each of them in a diagram-centric mode. Thanks to the **MagicDraw DSL Customization Engine** a set of **custom diagrams have been defined** (Fig 2): each custom diagram is derived from a base UML2 diagram. In the following list there is the associations between views and corresponding Plastic custom diagrams.

(i) **Requirements View** :
   a. Use Case Service Diagram (base: Use Case Diagram, dynamic)

(ii) **Services View**:
   a. Service Description Diagram (base: Class Diagram, static)
   b. Business Process Description (base: Activity Diagram, dynamic);

(iii) **Components View**:
   a. Service Specification Diagram (base: Class Diagram, static),
   b. Elementary Service Dynamics Diagram (base: Sequence Diagram, dynamic)

(iv) **Implementation View**:
   a. Component Design Diagram (base: Class Diagram, static);

(v) **Deployment View**:
   a. Service Deployment Diagram[1] (base: Deployment Diagram, static)
   b. Physical Mobility Pattern Diagram[2] (base: State Machine, dynamic)

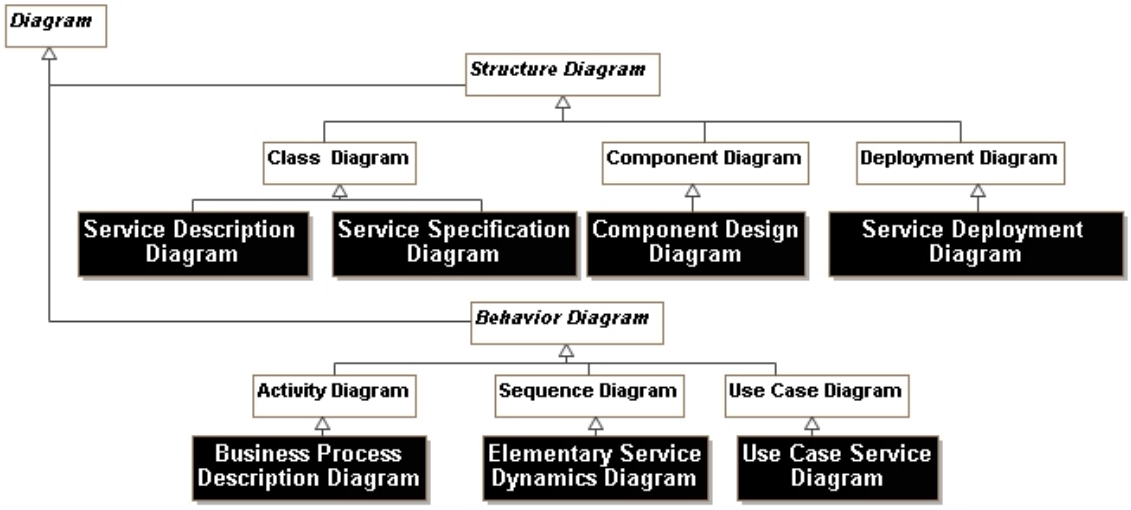


Fig 2: The taxonomy of PLASTIC custom diagrams

3.1 MagicDraw DSL Customization

DSL stands for Domain Specification Language. It is an advanced feature of MagicDraw modeling tool called **DSL Engine**. It allow to **customize the GUI** when a stereotypes belonging to Plastic Profile is assigned to a modeling elements: a new image and a suitably customized GUI for element specification can are provided. Moreover DSL Engine allows the hiding of unused modeling elements to drive the user during the service modeling.

3.1.1 Custom diagrams and custom toolbars.

The DSL customization allows the introduction of aforementioned custom diagrams. Each diagram has its own custom toolbar.
Each toolbar customization follow the following design principles:

(i) **Only allowed stereotyped modeling elements are added to the custom toolbar**;

(ii) **Allowed stereotyped elements are divided into button sets**, distinguished by the non functional quantitative information conveyed with stereotypes. Currently there are four possible sets:
   a. **Plastic related stereotyped elements** contained in a toolbar with the same name of the customized diagrams;
   b. **Performance related stereotyped elements** contained in *Performance Elements* toolbar;
   c. **Reliability related stereotyped elements** contained in *Reliability Elements* toolbar;
   d. **Mobility related stereotyped elements** contained in Mobility Elements toolbar;

(iii) **All customized diagrams contains the standard toolbar of the base diagram:** for example Use Case Service Diagram, derived from Use Case Diagram, contains the

[1] Physical Mobility Configuration Diagram is renamed in Service Deployment Diagram due to this custom diagram can be used disregarding the mobility requirements
[2] The Physical Mobility Pattern Diagram directly refers to the mobility requirement because of it has to be specified only in order to cope with mobility requirements.

standard *Use Case Diagram* toolbar. By default, the default toolbar is shown collapsed because a Plastic Service Modeler should use only stereotyped elements shown within customized toolbar sets.

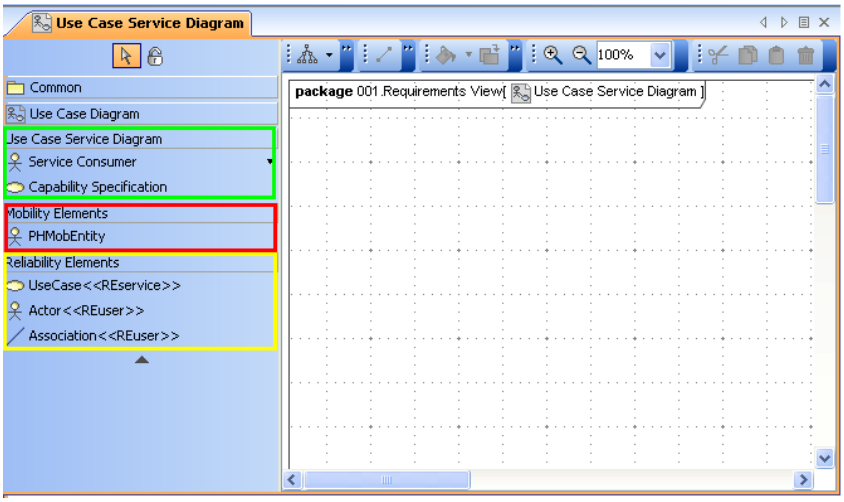


Fig 3: The customized toolbar of the Use Case Service Diagram

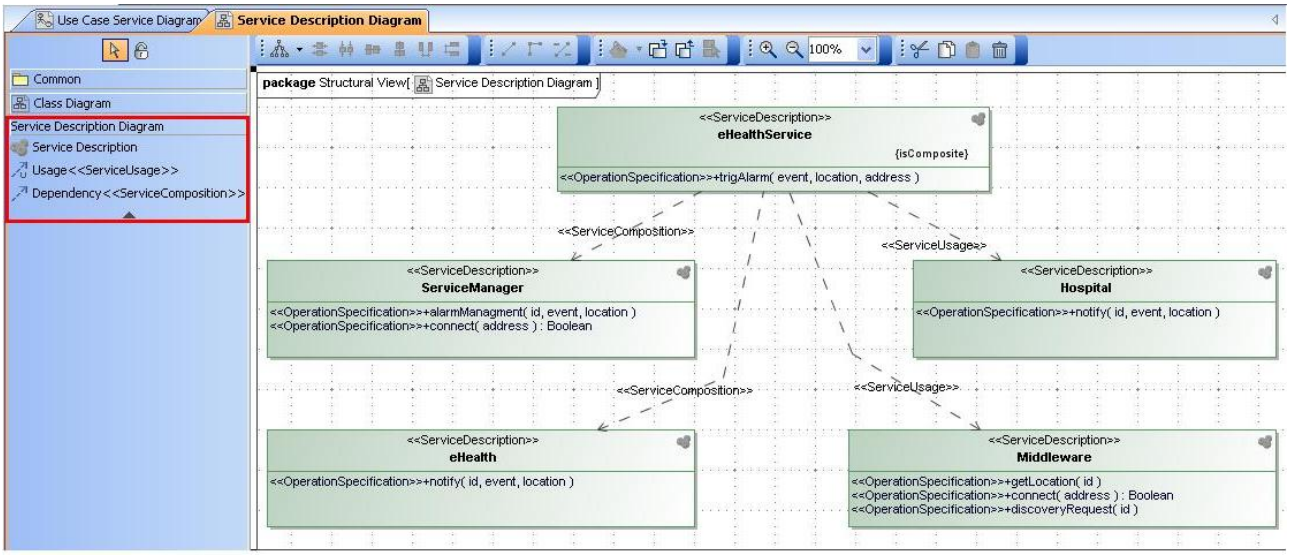


Fig 4: two examples of cutomized diagrams and their own customized toolbar: the Use Case Service Diagram and Service Description Diagram.

3.1.2 Customized context menus and customized specification dialogs

When a new Plastic-related stereotyped element is added to the model, it is possible to edit it

(i) through its **contextual menu** (right click on it within the *Containment* tree or on the shape displayed within diagram). As you can see from Fig 5 multiple stereotypes (but only allowed ones!) are shown within the contextual menu and can be applied on the same modeling elements by click on the menu items (checked = applied, unchecked = unapplied).

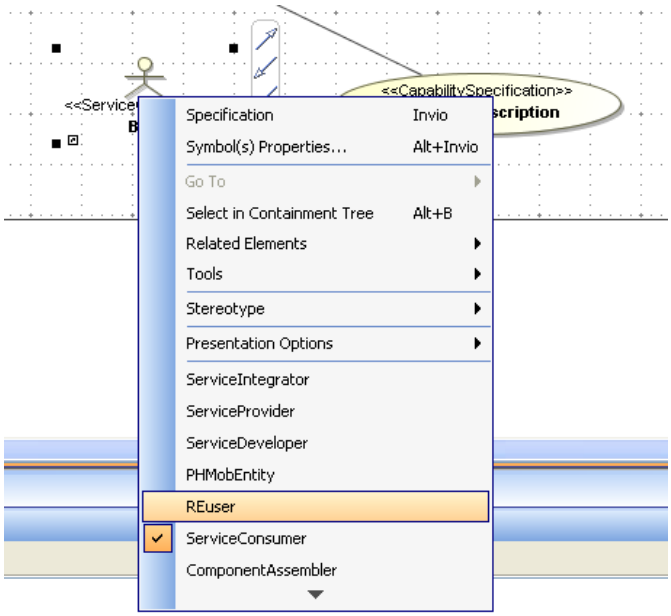


Fig 5: how to add quantitative additional information about reliability (REuser) or mobility (PHMobEntity).

(ii) Through its **customized Specification dialog** (select the model elements then click Enter or select *Specification* menu item in the contextual menu). If multiple stereotypes have been applied on the same modeling elements, the customization dialog adapts itself to show tagged values divided into sets based on non-functional additional information types (es. *Performance Attributes, Reliability Attributes…*)

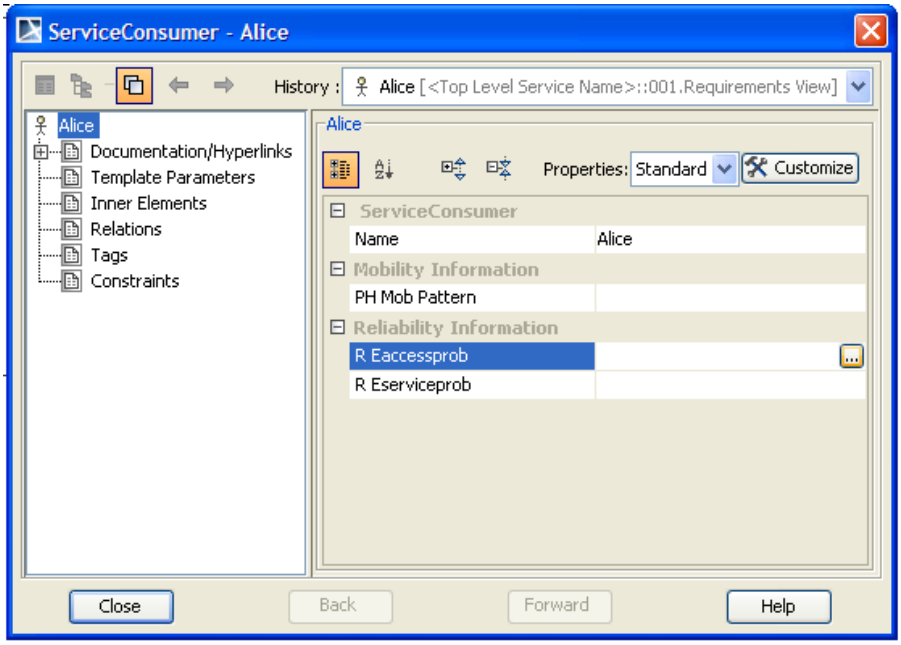


Fig 6: customized specification dialog for an Actor with multiple stereotypes applied: <<ServiceConsumer>> , <<REuser>> (reliability) and PHMobEntity (mobility).

The DSL customization **is a work in progress**. User's feedback is appreciated to improve the current DSL customization to speed up the service model creation and to hide error-prone useless GUI elements.

3.2 Performance, Reliability and Mobility references

Plastic Service Model includes additional information for sake of **performance**, **reliability** and **mobility** analysis. In the following sections we refers to these additional information to set what kind of diagram contains what kind of information: performance, reliability and mobility theory backgrounds are presented respectively in [**3**,**5**,**6**], [**7**,**8**] and [**6**]

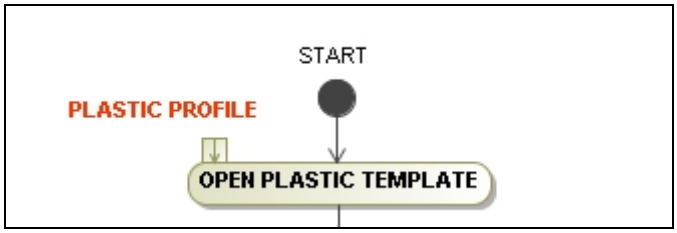
3.3 The Vacation Planning Service

The modeling example used in this manual in contained in ***Samples.Plastic.rar*** archive and described in [**1**]. After installation, this samples is contained within ***<magicdraw installation dir>/samples/Plastic***.
In the following sections we assume that the reader is skilled with the basic functionalities of MagicDraw® modeling environment. For user manuals end video tutorials, the reader can refer to the training material available within his own local installation or on the product website www.magicdraw.com (under the tutorial section.). While the example is introduced, chunks of an Activity Diagram are used to make the modeller aware of the right order of the modeling steps. Moreover, each step is further detailed with chunks of tree representing the model structure as result of the corresponding modeling activity (i.e. the actions displayed on the Activity Diagram).

3.3.1 The Vacation Planning Service Specification

The Vacation Planning Service specification is reported in **[TAWS]**: it has been suitably adapted to highlight the main modeling activities to define a complete PLASTIC Service Model.
John wants to plan a trip with his wife, to celebrate his new car. He starts planning the trip in his office with a laptop. He starts searching for a nice location: it must be close enough to where he lives (say, within 100 miles), by a lake and close to mountains. Moreover, John wants a nice and comfortable hotel, where they can rest and enjoy the fitness center. After finding the place, he makes a reservation for a room for the weekend, but since he has to run home, he does not wait for the confirmation from the hotel.
The confirmation message from the hotel arrives on John's mobile while he is returning home. As soon as John acknowledges the reservation, the hotel withdraws the agreed amount of money from his credit card. At home, he describes the almost planned trip to his wife and they start searching for good restaurants and places to see close to the chosen location. Again, they reserve tickets for a couple of museums, and also reserve a table in a nice restaurant by the lake for lunch on Saturday.
The day after, while waiting for his wife, John starts downloading on his mobile device the plan he had created using his laptop and the reservations done at home. Before leaving, they also need a map and a good service to identify the best itinerary to reach the place. Thus, they decide to exploit a simple and cheap map service offering the screen resolution supported by the mobile device, and asks the itinerary planning service for the fastest way to reach the target.
Given the characteristics of the display, the device automatically negotiates the best resolution that can be displayed and asks the credit card company to pay for it. Since the lowest resolution offered by the service is still too high for the capabilities of the display, the device needs a further service to filter the map.
Everything works fine, and John and his wife reach the hotel […]

3.3.2 Open the PLASTIC Service Template



Step 1.

Fig 7 illustrates the overall structure of a UML Plastic Service Model: this is a mandatory structure for any Plastic Service Model.

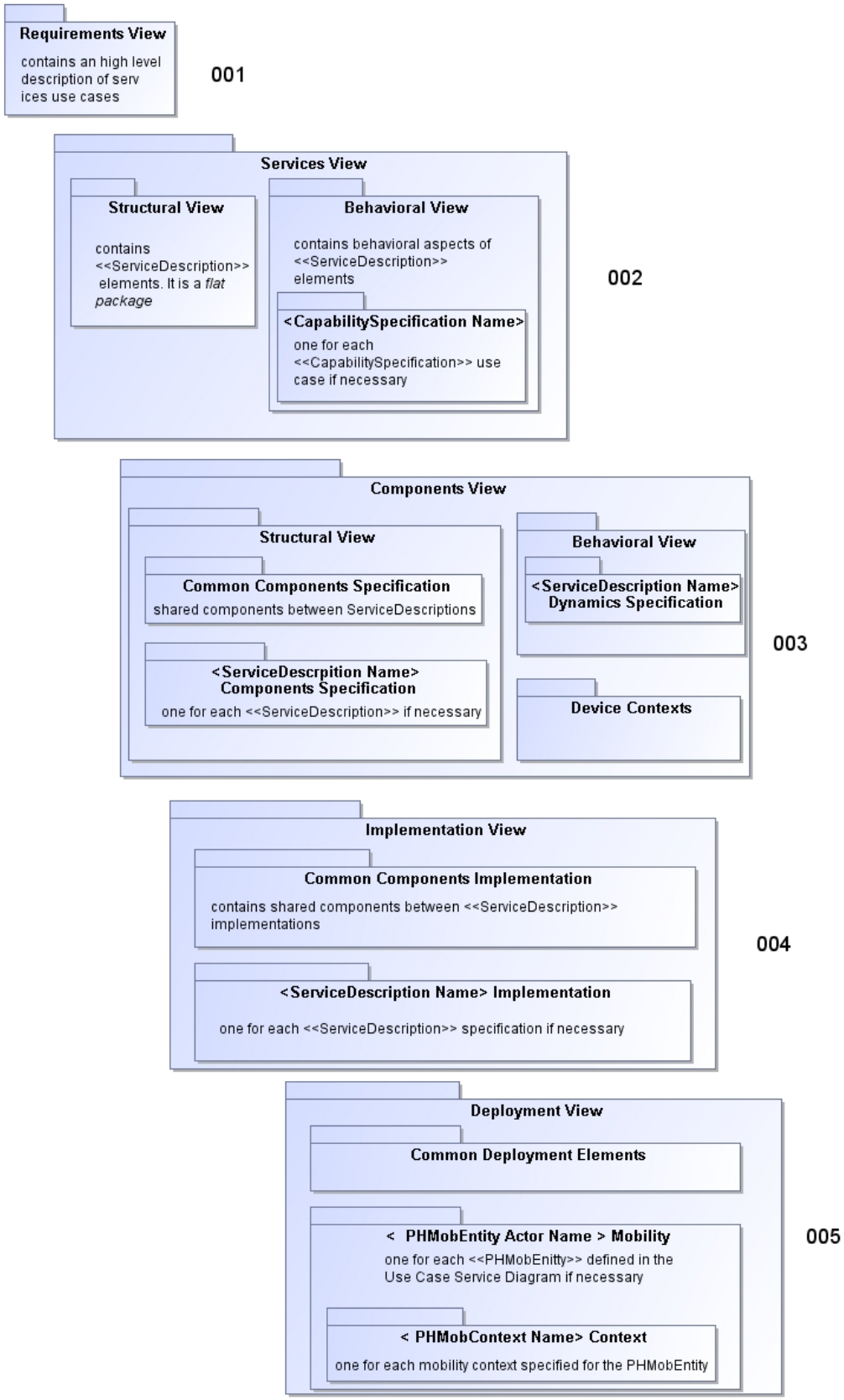


Fig 7: the overall structure of the UML PLASTIC Service Model.

To help any user to quickly have a well structured MagicDraw® project, a **Plastic Template** has been defined. This template is a predefined MagicDraw models with a suitable stereotyped package structure with all the required profiles (the Plastic Profile among them) already loaded and linked to the current model.
If you have correctly installed the archives provide, it is possible to choose this template by selecting:

1. ***File>New Project…>Project from Template>***
2. *select "**Plastic Template**" within Plastic folder.*
3. *click **OK***

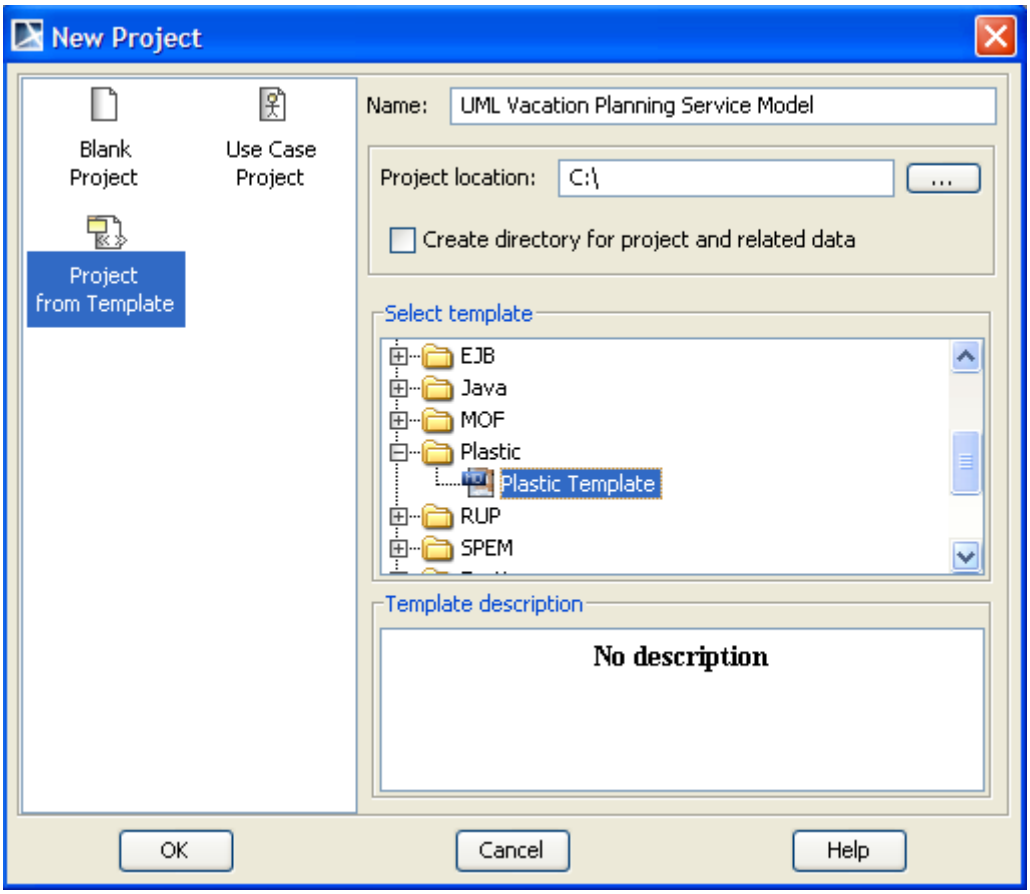


Fig 8: how to select the PLASTIC Template.

The resulting model structure is displayed in .
The package name surrounded by '<>' is a placeholder to be edited with the actual package name: for example <Top Level Service Name> have to be replaced with Vacation Planning Service.
The package stereotypes beside each package allow are used in order to tune the MagicDraw environment. They are used by the DSL Customization Engine® to define customized contextual menus, specification dialogs: for example, within a <<PLASTIC Service>> only five *kind* of packages can be added (see Fig 10)[3].

[3] It is worth to note that all the stereotypes displayed in this user manual belong to the PLASTIC Profile. On the other hand the stereotypes can be used both to model domain entities (the Core Specifications sub profile) or to ease the customization of the modeling environment (the Customization subprofile). See [PLASTICD2.2] and [PLASTICD2.3] for further information.

Fig 9: the default package structure displayed within MagicDraw.

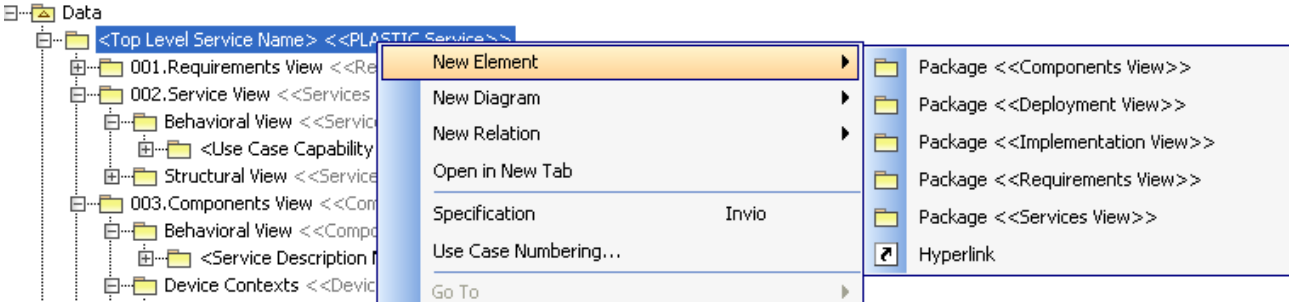


Fig 10: All the allowed views of the PLASTIC Service

In the following sections, all the steps required to specify the UML-side of a PLASTIC Service are described. **Each view is strictly related to the previous ones**: we recommend the readers to f**ollow a *waterfall* modeling activity**: f**irst the requirement view has to be completed then the service view can be specified and so on**. **Moreover, when required, the structural (sub)view takes precedence on the behavioural one**: the model elements used in the dynamic specification (i.e. lifelines, call messages or actions) usually refer to structural model elements (i.e. interfaces or components and their own operations)

3.3.3 Vacation Planning Service Model

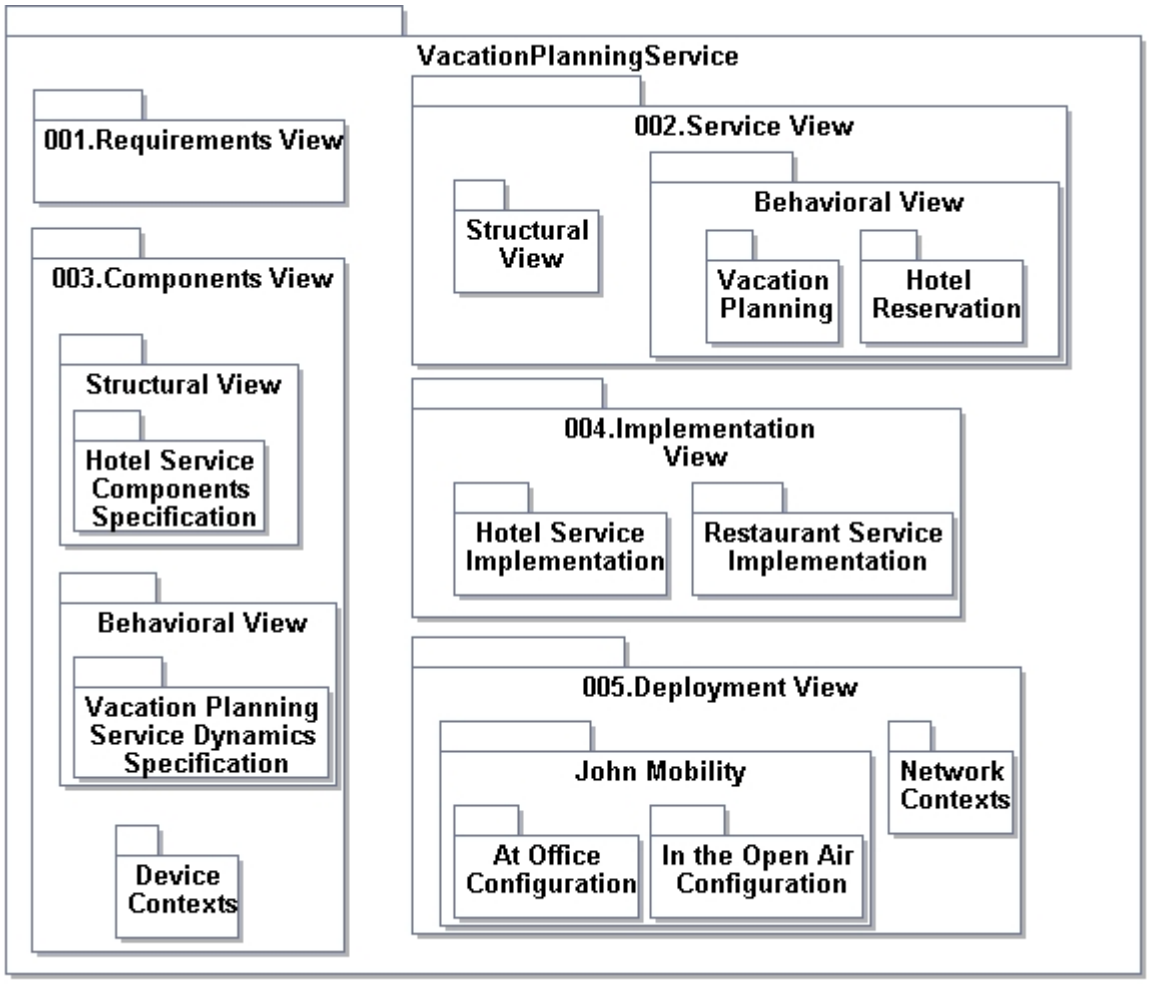
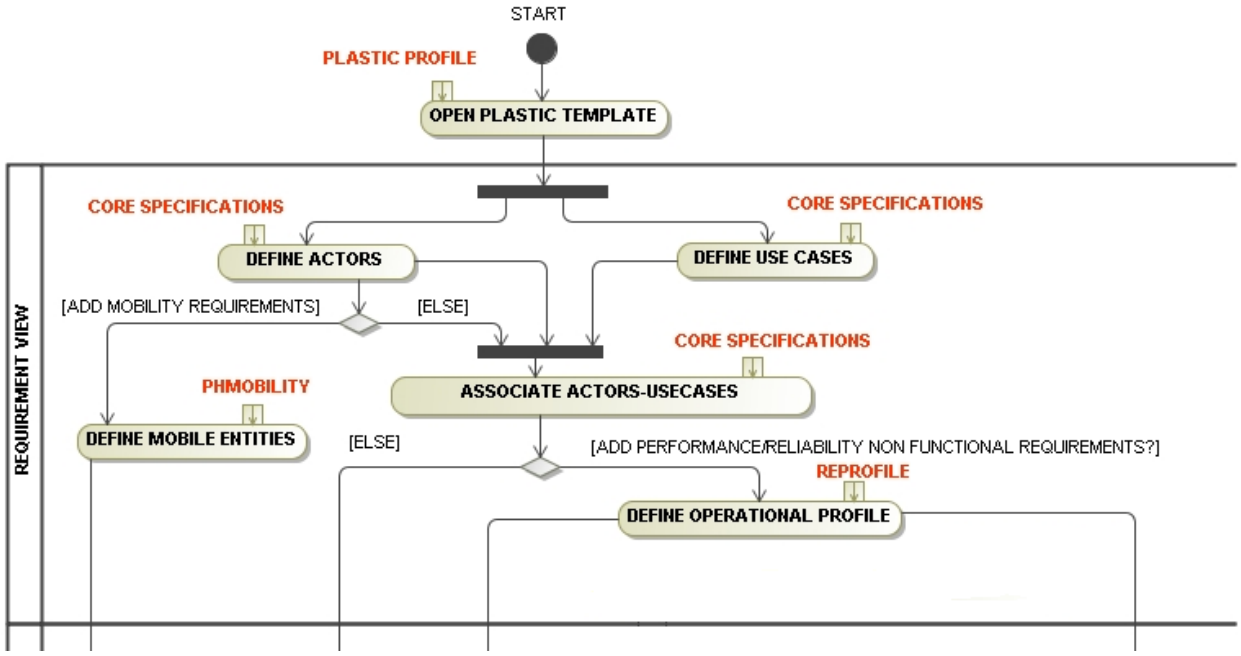


Fig 11: The VPS Service Model overall structure.

In *Fig 11* the overall service model package structure is shown.

3.3.4 The Requirement View



Step 2.

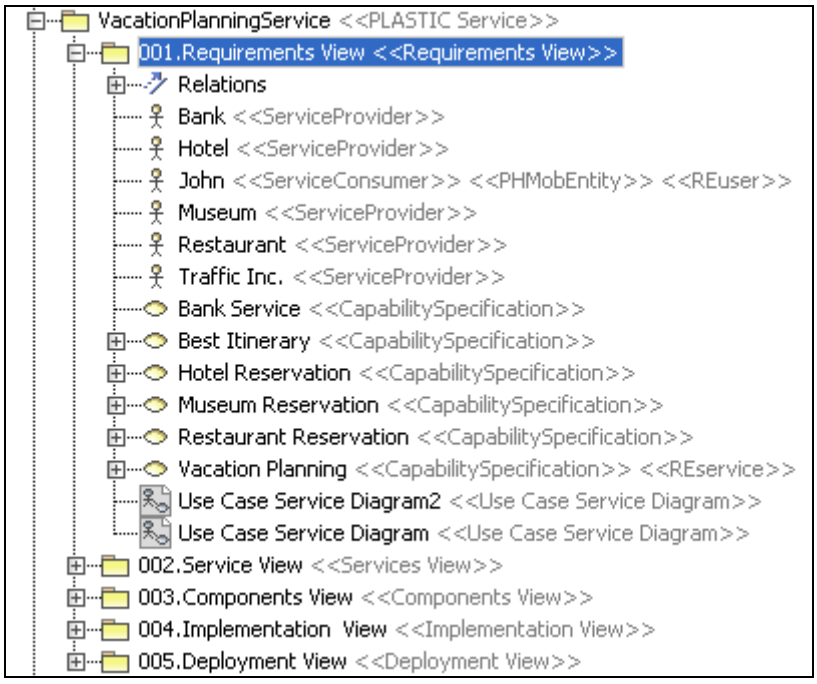


Fig 12: The VPS Requirement View structure.

The description of a PLASTIC application starts with **the Requirement View** specification (i.e. a set of structured UML model elements listed in the above tree) that is *graphically represented* by means of a set of **Use Case Service Diagram (UCSD,** an extension of the standard UML2 Use Case Diagram).

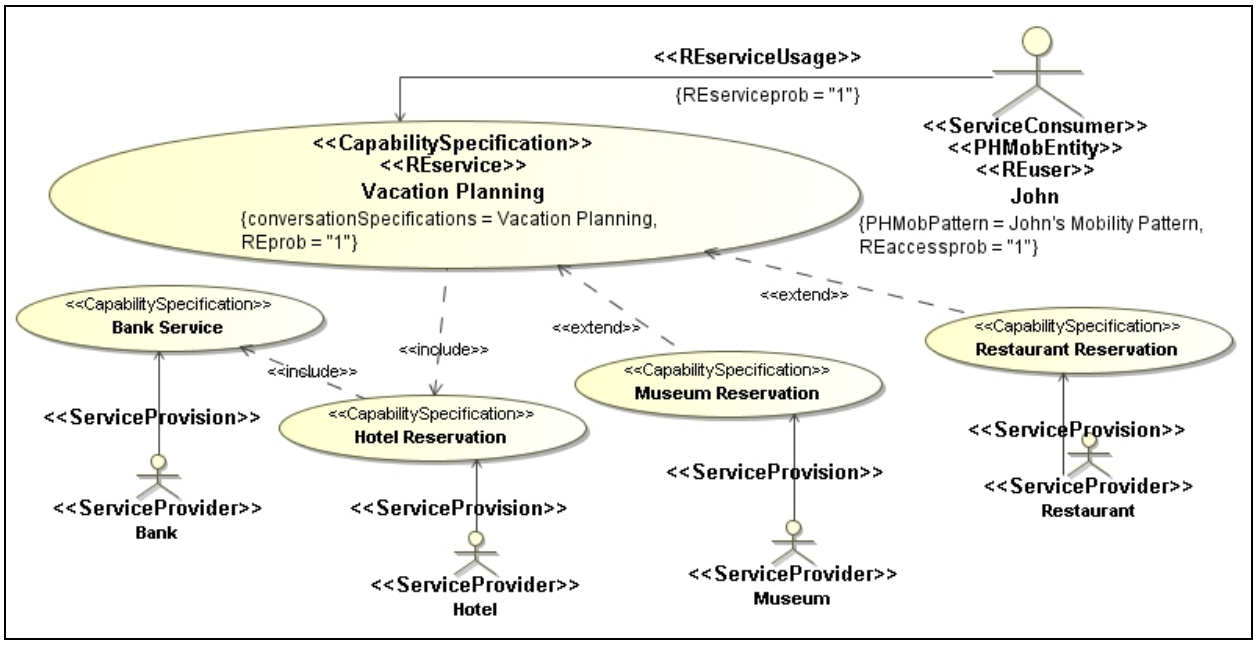


Fig 13: VPS Use Case Service Diagram.

In our case a *ServiceConsumer* (i.e. John) interacts with one *CapabilitySpecifications* (i.e. Vacation Planning UseCase) that is able to provide the desired service by means of a suitable orchestration (specified by the *ConversationSpecification*s tag) of changing sets of related services (i.e. *CapabilitySpecifications* that can <<*extend*>> the top service).

> **!** *The ConversationSpecification tag value is an Activity that will be specified at the Service View::Behavioral View.*

Moreover **the Requirement View** (graphically represented by one or more UCSD) **also some additional information to derive Analysis Models:**

- The *REuser*, *REserviceUsage* and *REservice* tags [PLASTIC D2.2] to define the operational profile for sake of performance and reliability analysis:

- The Physical Mobility Entity (*PHMobEntity,* e.g. John*)* with its own Mobility Pattern (see
- Fig *13*).

The model elements can be added on the model tree using the contextual menus

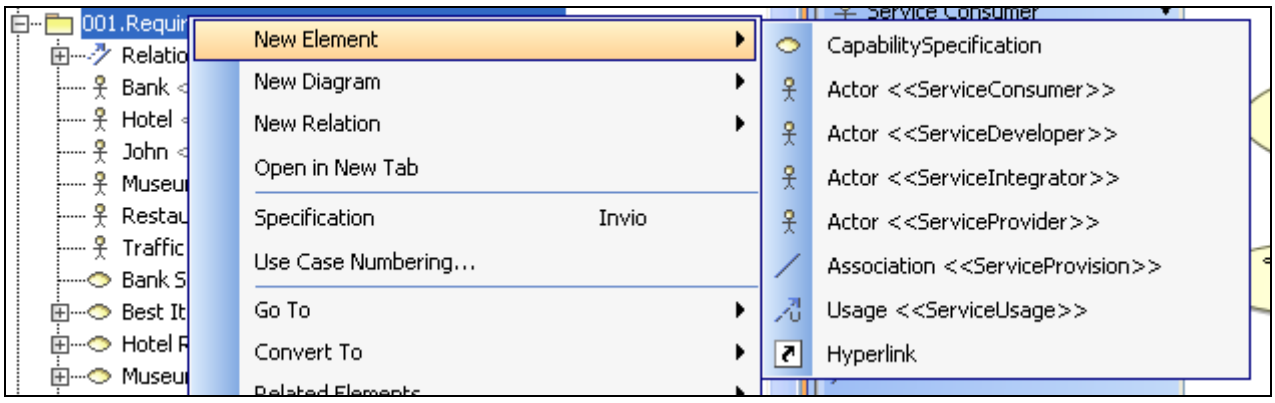


Fig 14: Using the contextual menus to define the Requirement View.

or on the Use Case Service Diagram using the **customized** visual editing capabilities of MagicDraw (e.g. **customized toolbar**)

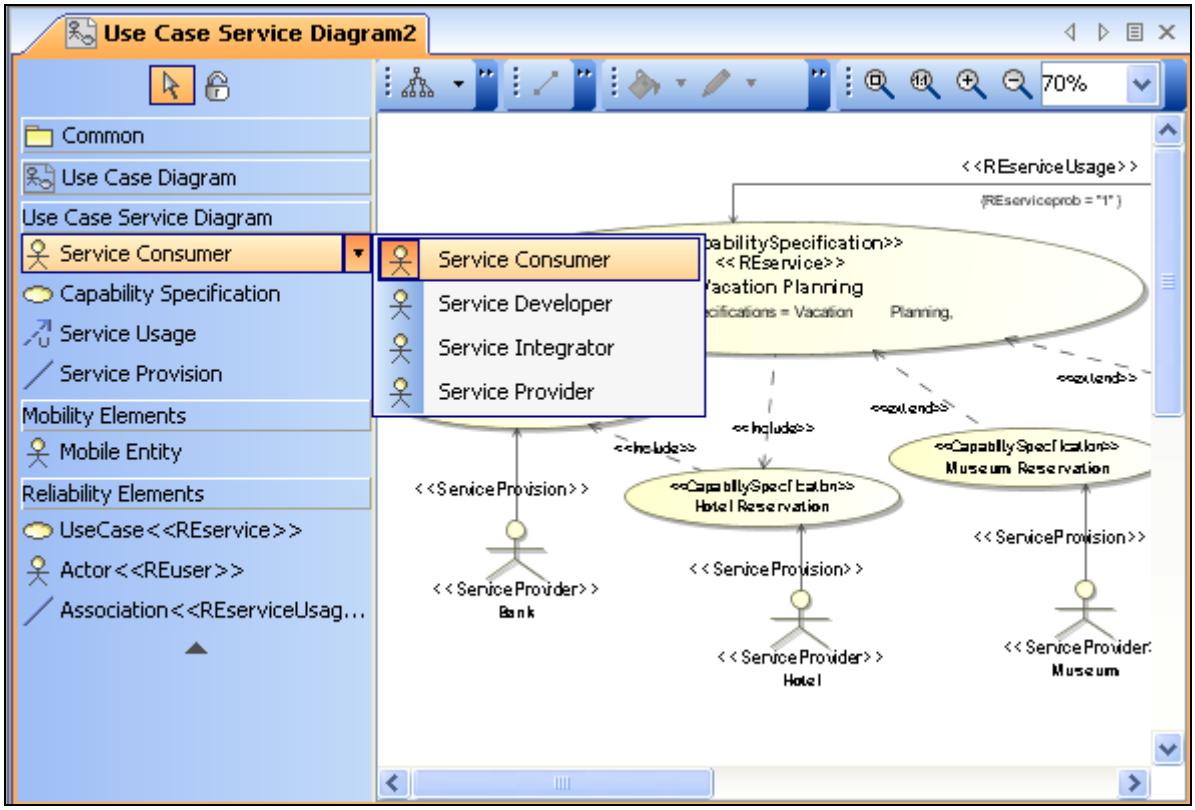


Fig 15: Using the custom diagram and its customized toolbar to define the Requirement View.

3.3.4.1 Model elements used at Requirement View.

In the following table, the stereotypes used at Requirement View are listed.
When not specified, the semantic is specified in [PLASTICD1.2][PLASTICD2.2].

Model Element	Base Metaclass	Target	Tags	Semantic
Service Consumer	Actor	Service Modeling	-	-
Service Provider	Actor	Service Modeling	-	-
Service Developer	Actor	Service Modeling	-	-
Service Integrator	Actor	Service Modeling	-	
Capability Specification	Use Case	Service Modeling	ConversationSpecification: Interaction	A ConversationSpecification represent the services interaction at business level.
REuser	Actor	Operational Profile	REaccessprob: float	REaccessprob represents the probability that a certain type of user accesses to the system.
REserviceUsage	Association	Operational Profile	REserviceprob:float	REserviceProb represents the probability that the type of user, once accessed, requires a certain service.
REservice	Use Case	Operational Profile	REprob:float (derived)	REprob represents the probability that a certain service is required. For each REservice, REprob is obtained as the sum (over all REuser) of the products between REaccessprob and REserviceprob. See **[Cortellessa RE]**
PhMobEntity (Mobile Entity)	Actor	User Mobility Modeling	PHMobPattern:StateMachine	See **[Mobility]**

Table 1: model elements used in Requirements View

3.3.5 The Service View

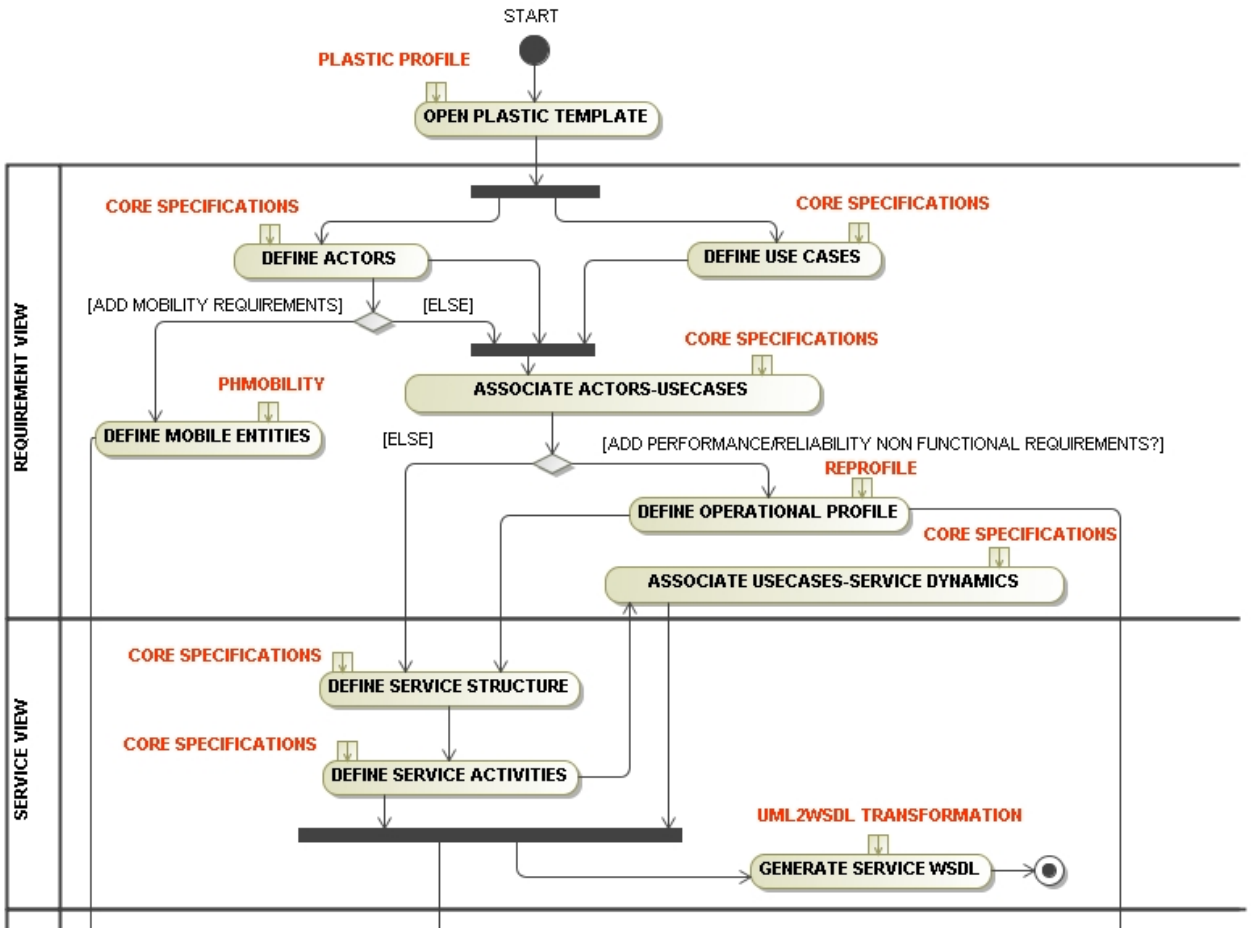


Fig 16: The VPS Service View.

3.3.5.1 The Service View: Structural View

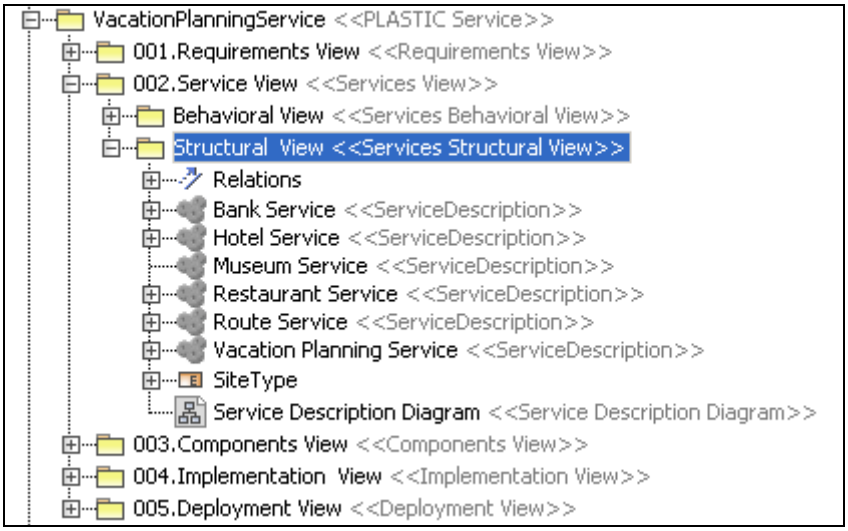


Fig 17: The VPS Service View::Structural View

Once the Requirement View has been produced, the development proceeds with the definition of the services that build up the PLASTIC application being modeled. Such specifications are given from both structural and behavioral perspectives. In particular, a Structural View is given by means of **Service Description Diagrams (SDescrD)** that show the *ServiceDescription*s (e.g. HotelService, RestaurantService) that may be combined on demand (*ServiceComposition* or *ServiceUsage*[4] dependency from client to supplier services) and collaborate to provide the required service (the top level Vacation Planning Service) to the nomadic user (e.g John).
The key concept is *ServiceDescription* which is the base structural unit for the description of PLASTIC applications at service level. It is a stereotype extending the UML2 *Interface* meta class. It provides some *OperationSpecification*s that, together, define what the user can request from its PLASTIC enabled device (e.g. John's laptop or PDA).
It is worth to note how, for example, the DSL Engine allows to specify a different concrete syntax for the Service Description stereotyped Interface.

[4] A *ServiceComposition* dependency is used when the supplier service has to be modelled then implemented as full PLASTIC Service while *ServiceUsage* dependency is used when the supplier service already exists.

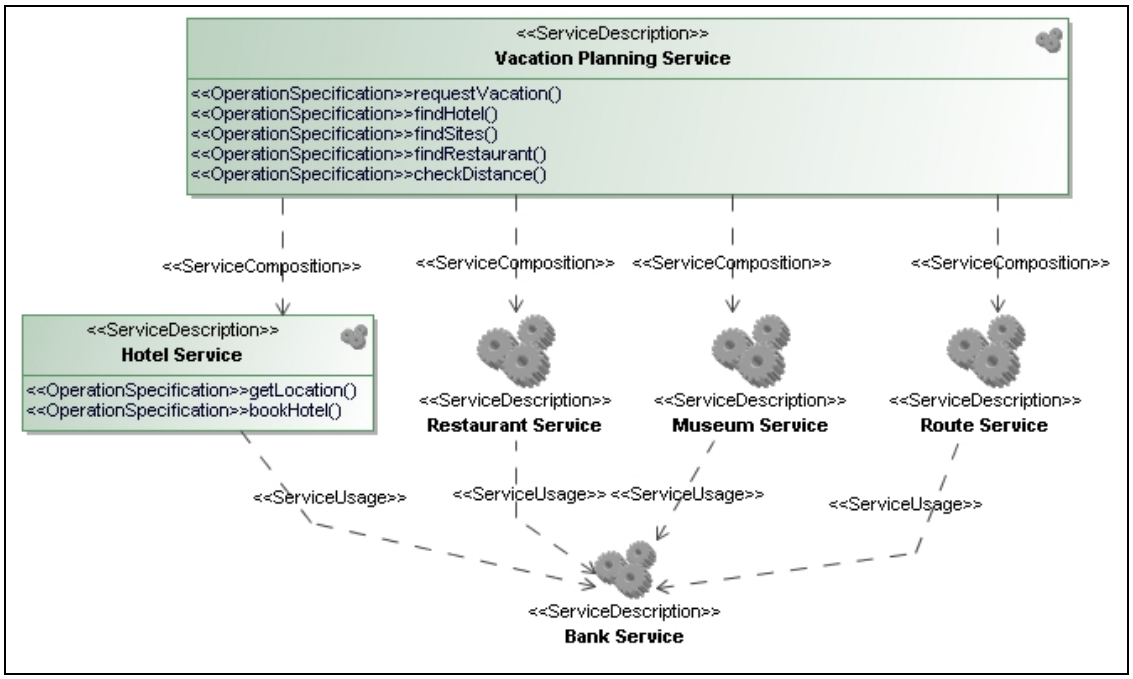


Fig 18: The VPS Service Service Description Diagram.

Once again, the **model elements or customized diagrams** can be added on the model tree using the contextual menus (right click on the tree node *Structural View*), or on the Service Description Diagram using the **customized** visual editing capabilities of MagicDraw (e.g. **customized toolbar**)

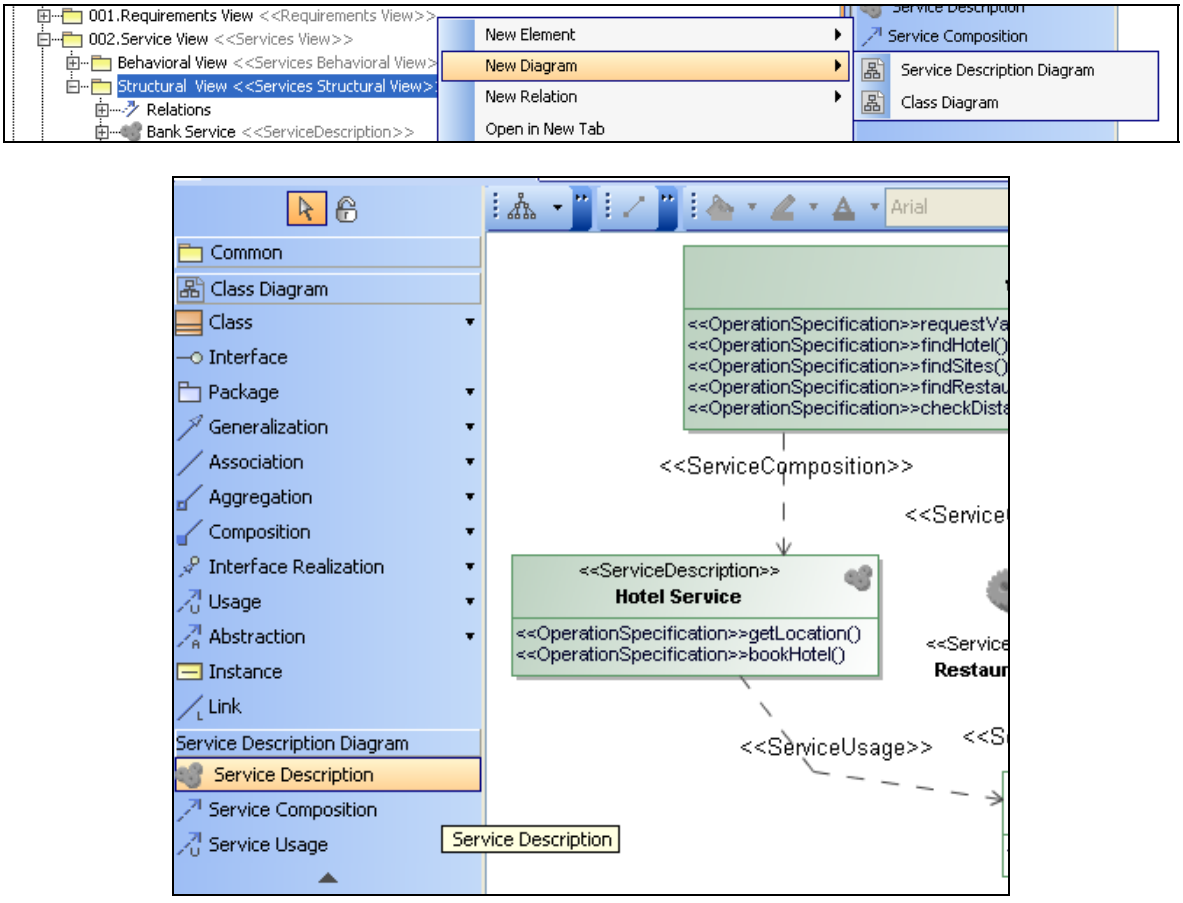
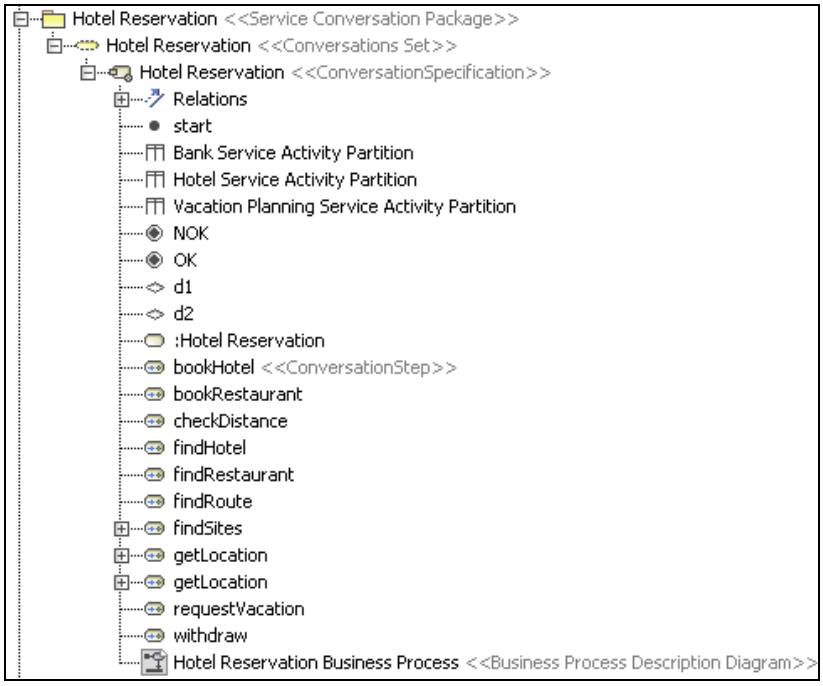


Fig 19: Using the contextual menus or the customized toolbar to define the Service View.

3.3.5.2 The Service View: Behavioral View



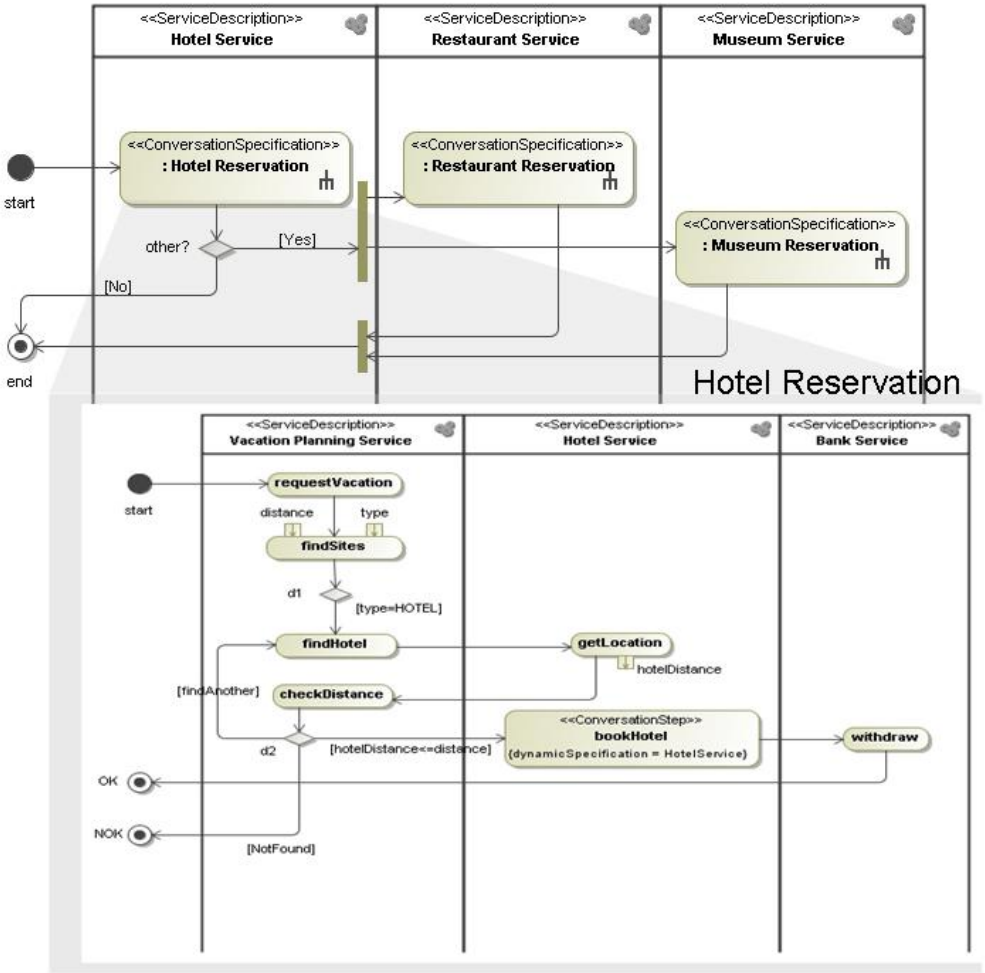


Fig 20: The BPDD for the ConversationSpecification associated to the Hotel Reservation Use Case

Once all services are defined (i.e. the static description), a number of **business process descriptions** (i.e. the behavioural description) have to be provided. Each Business Process Description describes the activities (i.e. service orchestration) between the *ServiceDescriptions* identified in the SDescrD.
In particular, **for each *CapabilitySpecification*** (e.g. HotelReservation UseCase specified at the Requirements View), **a *Business Process Description Diagram* (BPDD) has to be specified to describe the interactions among the involved services**.
A possible Vacation Planning overall orchestration includes at least an Hotel Reservation (i.e. the <<include>> relationship between Vacation Service and Hotel Reservation Use Cases). Then the *ServiceConsumer* John can extend the basic service with a Restaurant and Museum Reservation services (i.e. <<extend>> relationship on UCSD, see) .
The detailed *ConversationSpecification* for each requested service is modeled by means of another BPDD: for each Activity Partition, each Action invokes the homonymous *OperationSpecification* of the *ServiceDescription* in the caption. A service can invoke other services (i.e. it can exist transitions between swim lanes) if and only if a *ServiceUsage* or *ServiceComposition* relationship exists between the involved *ServiceDescriptions* (this is an example of cross-diagram validation rule).

3.3.5.3 Model elements used at Service View.

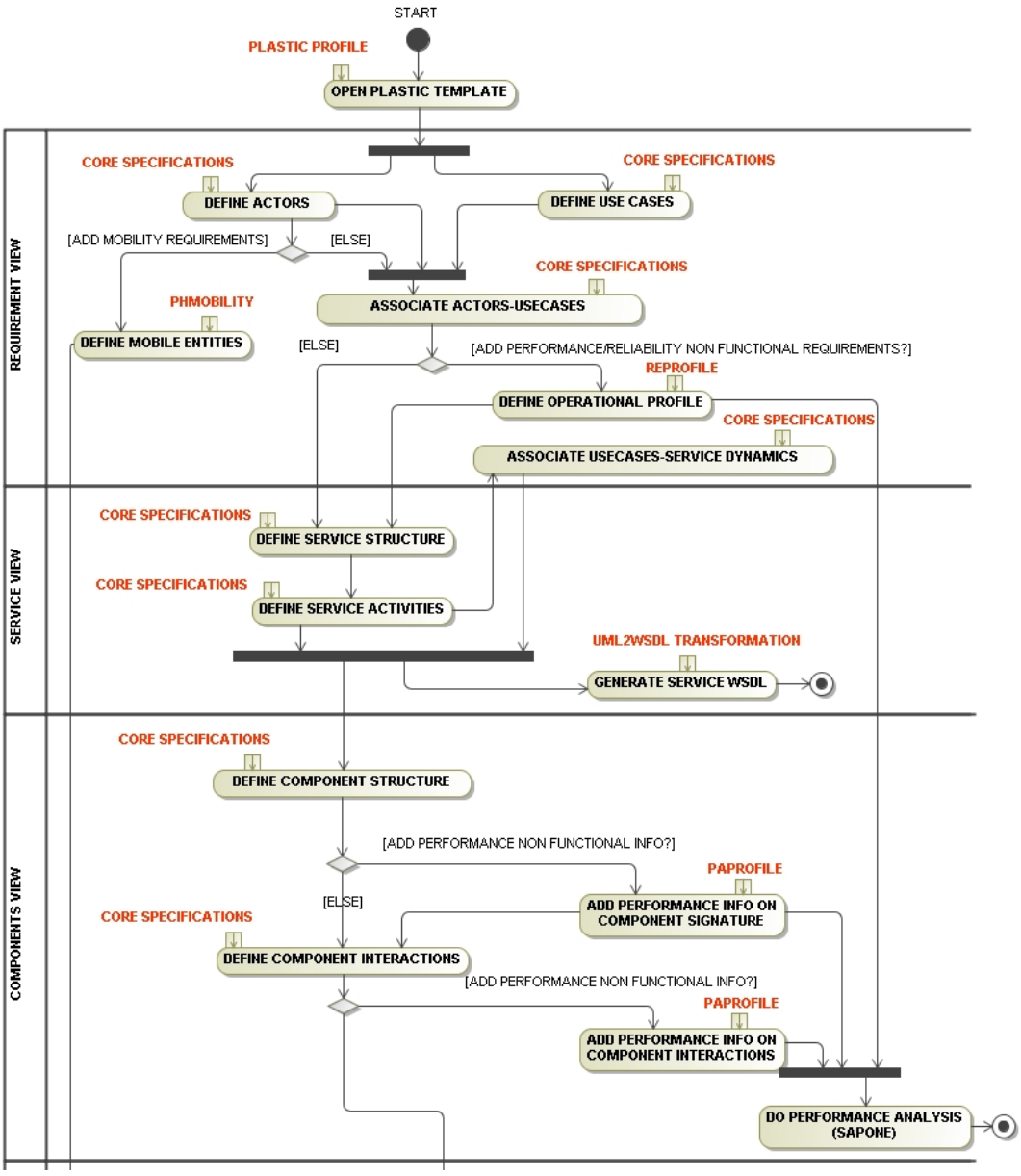
In the following table, the stereotypes used at Service View are listed.
When not specified, the semantic is specified in [PLASTICD1.2][PLASTICD2.2].

Model Element	Base Metaclass	Target	Tags	Semantic
Structural View				
ServiceDescription	Interface	Service Modeling	-	-
OperationSpecification	Operation	Service Modeling	-	-
ServiceComposition	Dependency	Service Modeling	-	-
ServiceUsage	Dependency	Service Modeling	-	-
Behavioral View				
Conversation Specification	Activity	Service Modeling	-	-
CallOperationAction	Call Operation Action	Service Modeling	Operation: OperationSpecification	-
ConversationStep	Call Operation Action	Service Modeling; Performance Analysis	dynamicsSpecification: Interaction	Its value is a reference to the Interaction representing the software behaviour at components level.

Table 2 model elements used in Services View

3.3.6 The Components View

Fig 21: The VPS Component View



Step 3

In general, a service can be implemented by one or more software components and, in turn, a software component can be used to implement one or more services. The PLASTIC Profile provides modeling constructs aimed at describing the component based software architecture that implements a given service and their interactions. Such descriptions are organized in a ***Component View*** **and distinguished in** ***Structural View*** **and** ***Behavioral View***, respectively.

3.3.6.1 The Components View: Structural View and Device Contexts

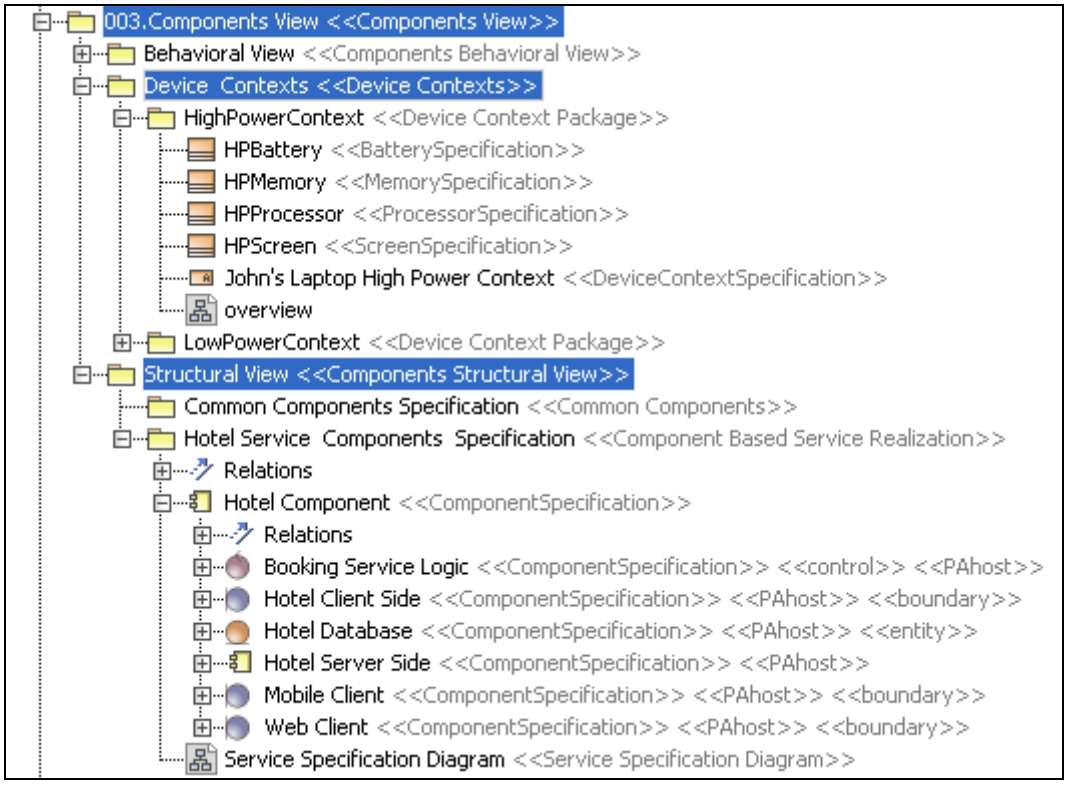


Fig 22: The Service Specification Diagram for the Hotel Service.

The PLASTIC profile proposes the use of ***Service Specification Diagrams*** **(SSD)** for defining the components implementing the service represented by a certain *ServiceDescription*. Such diagrams are expressed as extended UML2 Class Diagrams and a number of new modeling constructs are provided. The *ServiceRealization* stereotype is introduced to link *ServiceDescription* stereotyped interface and *ComponentSpecification* stereotyped components. It describes how services are implemented in terms of software components. Moreover, by means of the SSD the designer can specify the contexts in which the service will be able to work after adaptation. In particular *DeviceContextSpecification* elements are used to describe possible devices (e.g. John's laptop, mobile or car navigator): each tag refers to an available resource specification of the Resource package. The *DeviceContextSpecification* is then linked to adaptable services by means of *ServiceAdaptation* relationships. The SSD in *Fig 23* shows the component-based software architecture of the HotelService: it is a client-server application where, following the MVC architectural pattern, the model (HotelDatabase entity) and the control (Booking Service Logic control) are placed on the server side (Hotel Service Side), whereas the client-side (Hotel Client Side boundary component) provides the service interface on two possible kinds of PLASTIC enabled devices (Mobile Client and Web Client).

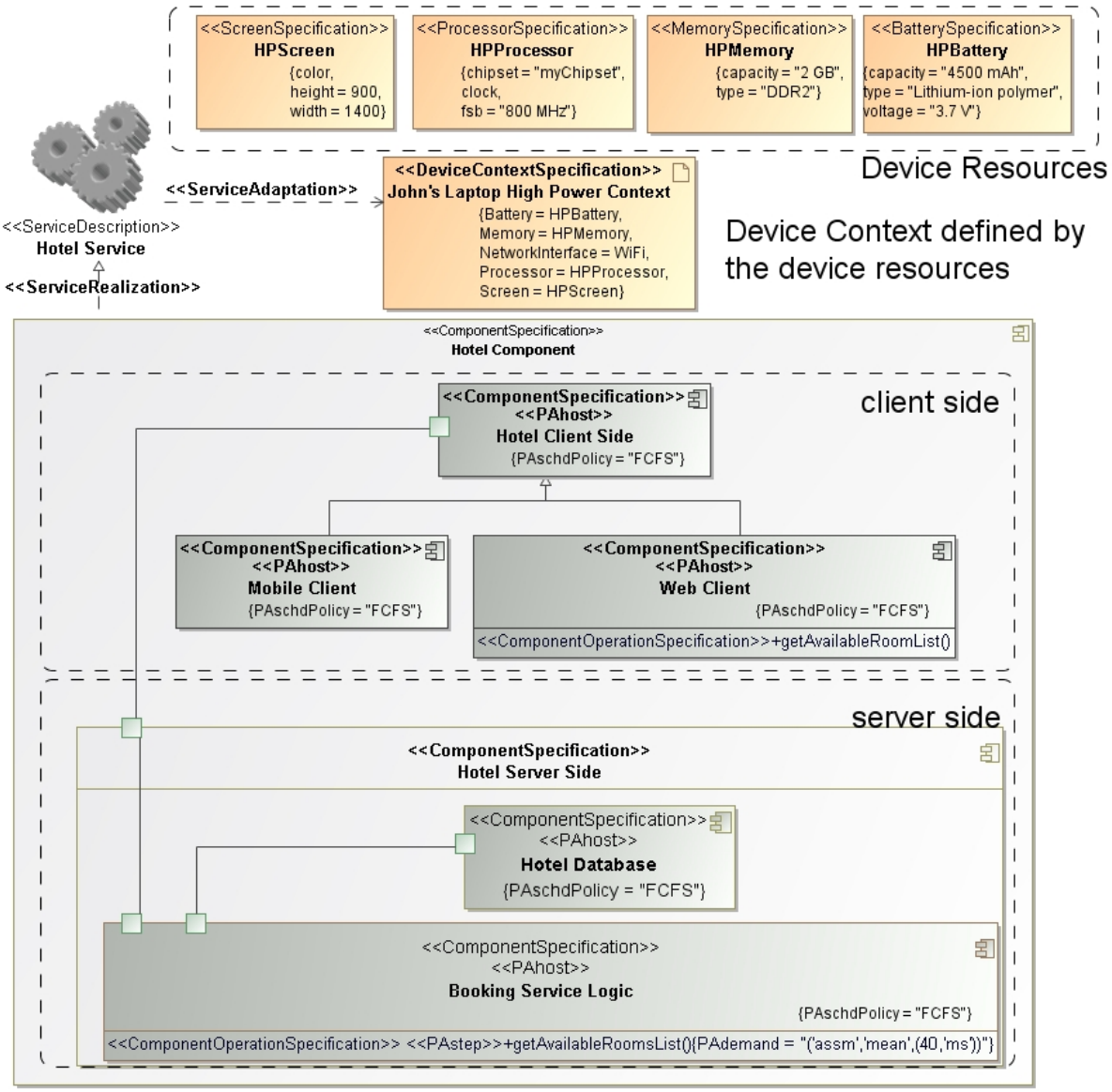


Fig 23: The Service Specification Diagram for the Hotel Service.

3.3.6.2 The Components View: Behavioral View

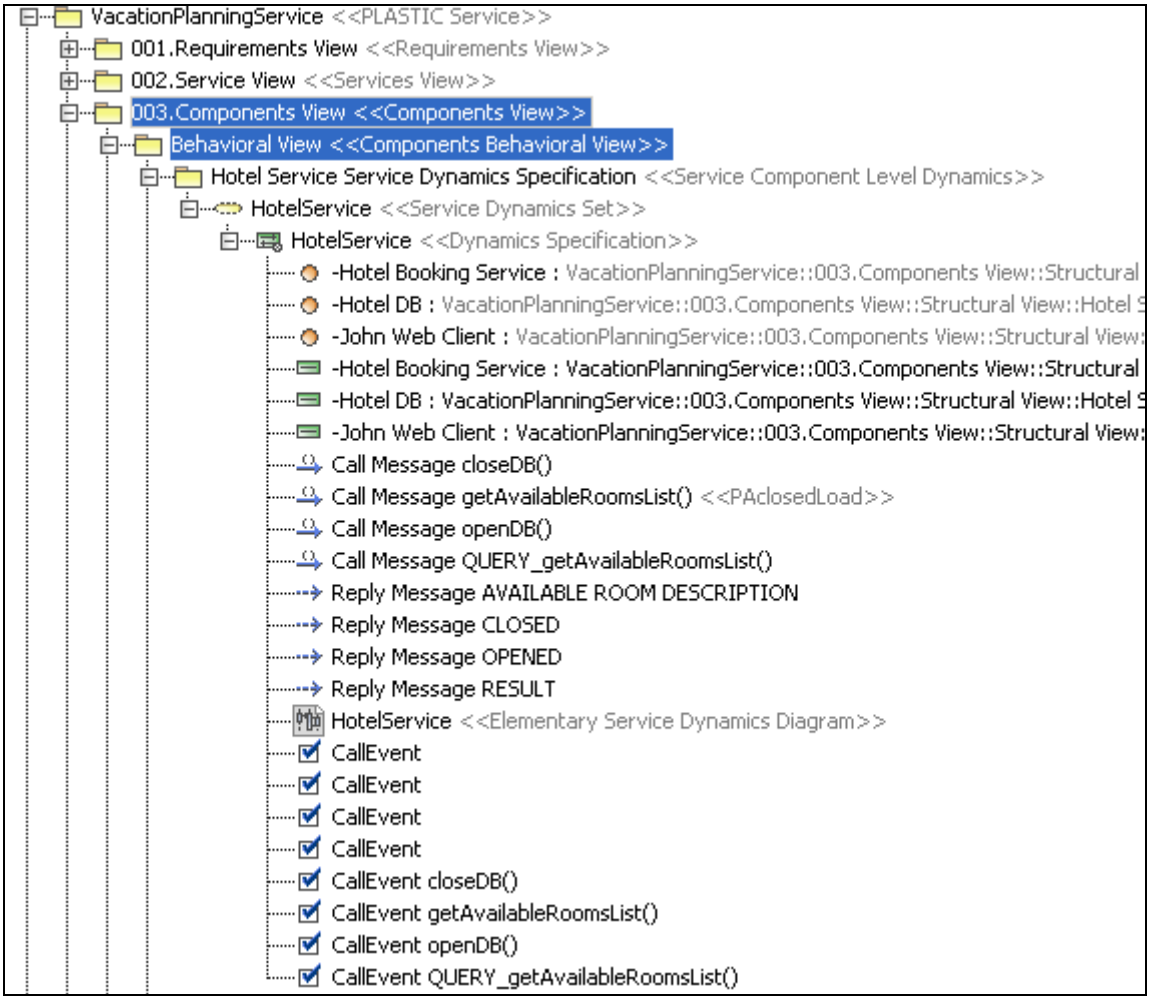


Fig 24: The Component View::Behavioral View of the VPS

Once the components implementing the service being modeled have been defined, their interactions have to be specified. The PLASTIC profile provides the designer with *Elementary Service Dynamics Diagrams (ESDD)* which are given as an extension of UML2 Sequence Diagrams to model the interactions among the involved components specified in the structural view by means of *ComponentSpecification* elements.

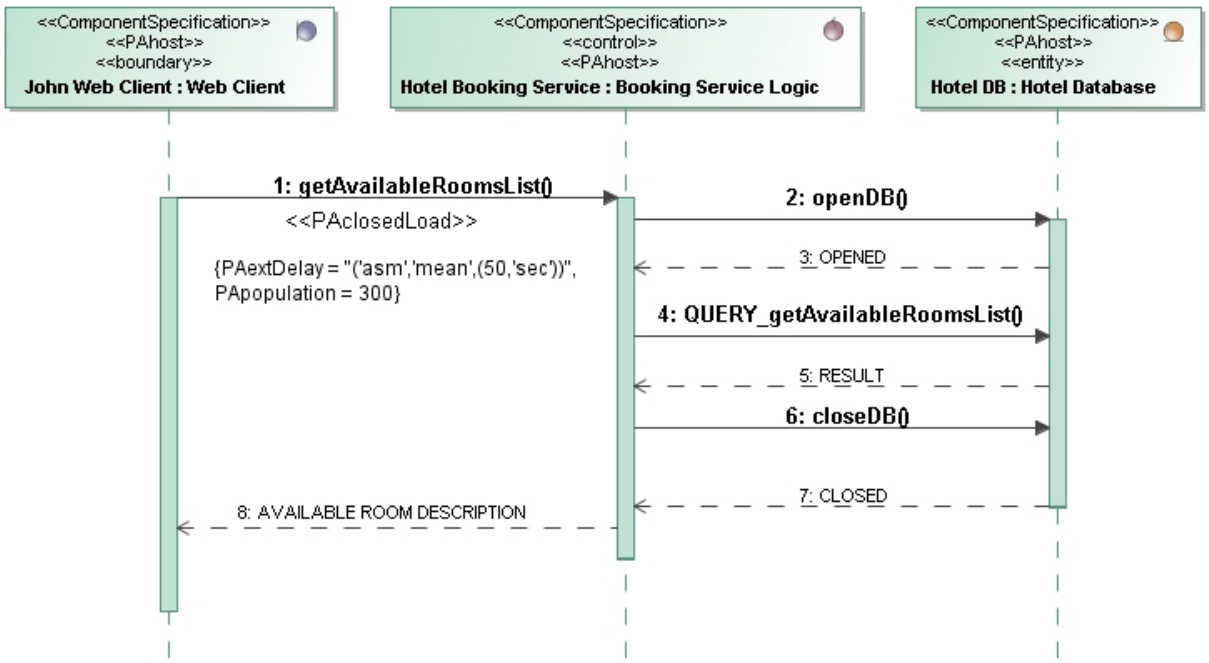


Fig 25: The Elementary Service Dynamics Diagram for the Hotel Service

3.3.6.3 Model elements used in the Components View.

In the following table, the stereotypes used at Components View are listed.
When not specified, the semantic is specified in [PLASTICD1.2][PLASTICD2.2].

Stereotype	Base Class	Target	Tags	Semantic
Component Specification	Component	Service Modeling	-	-
Component Operation Specification	Operation	Service Modeling	-	-
Service Realization	Realization	Service Modeling	-	-
Service Adaptation	Dependency	Service Modeling	-	-
Device Context Specification	Artifact	Device Context Modeling	-	-
Battery Specification	Artifact	Device Context Modeling	-	-
Memory Specification	Artifact	Device Context Modeling	-	-
Processor Specification	Artifact	Device Context Modeling	-	-
Screen Specification	Artifact	Device Context Modeling	-	-
PAhost	Component	Performance Modeling	-	If the component is labelled with this stereotype its queue has a *type*. If a component is not labelled, than it will map to a delay resource without queue and with an infinite number of instances. [SAPONE], [SPT]
PAstep	Operation, Message	Performance Modeling	{Padelay=('assm','mean',(*value*,'*unit time*'))} {Pademand=('assm','mean',(*value*,'*unit time*'))}	The PAdelay is placed on the first message outgoing from delay components. The PAdemand is placed on the first message outgoing from active components. The remaining time to end the process started from the first message of the Sequence Diagram. [SAPONE], [SPT]
PAclosedLoad	Message	Performance Modeling	{Papopulation=*population quantity*, PaextDelay=('assm', 'mean' ,(30,'ms'))}	It have to be placed on the first message of the Elementary Service Dynamics Diagram.This stereotype denotes the closed workload of the Sequence Diagram, in particular the population and the delay introduced from the population. [SAPONE], [SPT]

Table 3 model elements used in Components View

3.3.7 The Implementation View

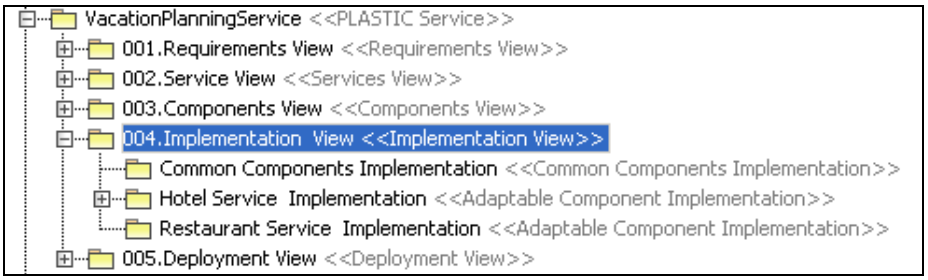
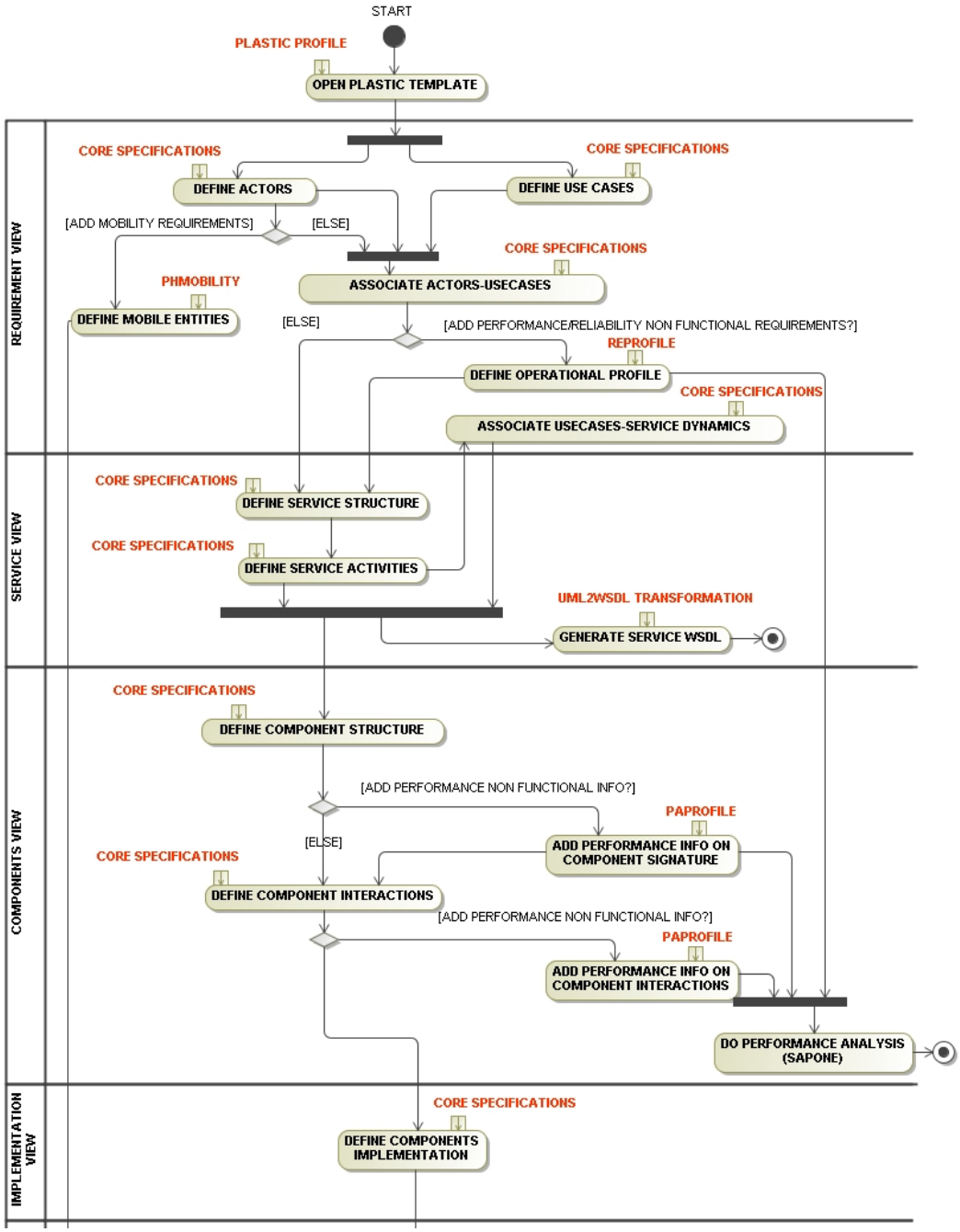


Fig 26: The VPS Implementation View.



Step 4

In order to have a comprehensive description of the PLASTIC service being developed, each component implementing it needs to be specified at a lower level of abstraction. Such descriptions give place to the *Implementation View* which consists of ***Component Specification Diagram (CSD)*** that describe the classes implementing the components specified in the *Component View.* The *AdaptableClass* modeling element is used to distinguish the classes which are adaptable according to specified resource annotations [chamaleon].
We provide just a trivial implementation view of the Hotel Booking Service. It is a PLASTIC Service then it is a client-server application distributed over B3G networks. The client-side need to provide the desired service both on a Web (*HotelWebClient*) and Mobile Client (*HotelMobileClient*): the *ComponentImplementationSpecification* are the corresponding deployable artifact on the John's PLASTIC enabled device (laptop or mobile) where he is able to see the hotel list and the his reservation result. The *Adaptable Class*es with their *ContextAttribute*s and *AdaptableMethod*s are aware of their own running device (the DeviceContext which the HotelService has to adapt to, see the SSD in *Fig 23*). On the server side (*Hotel Server Side*) the business logic is implemented by the HotelServiceSkeleton and HotelBooking *AdaptableClass*es. These two classes are in charge to manage the dataflow from and to the HotelDatabase entities and send them back to the Hotel Client Side according to the available resources on the receiving device.

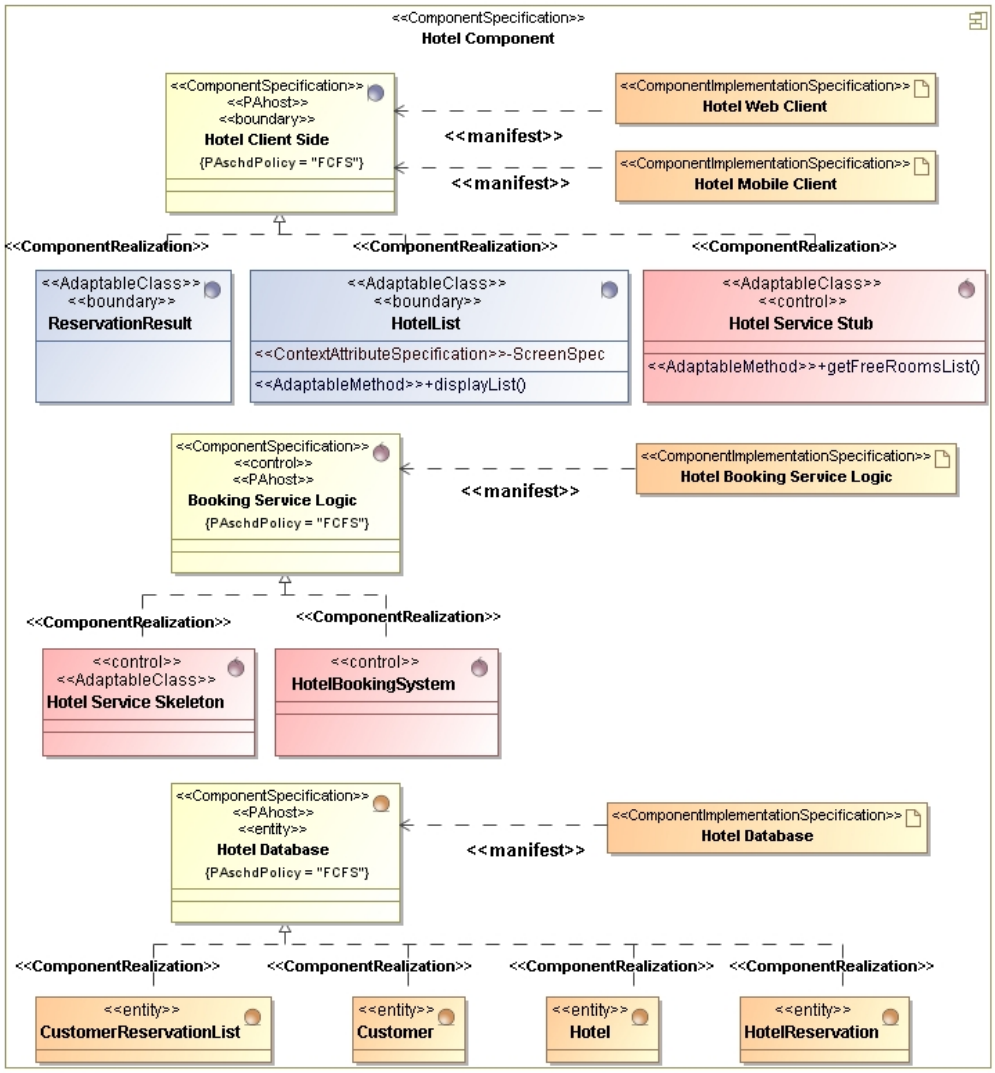


Fig 27: The VPS Component Design Diagram.

3.3.7.1 Model elements used at Implementation View.

In the following table, the stereotypes used at Implementation View are listed.
When not specified, the semantic is specified in [PLASTICD1.2][PLASTICD2.2].

Stereotype	Base Class	Target	Tags	Semantic
Adaptable Classes	Class	Implementation Modeling	-	-
Adaptable Method	Operation	Implementation Modeling	-	-
Component Realization	Realization	Implementation Modeling	-	-
manifest	Dependency	Implementation Modeling		
Component Implementation Specification	Artifact	Implementation Modeling	-	-

Table 4 model elements used in Implementation View

3.3.8 The Deployment View

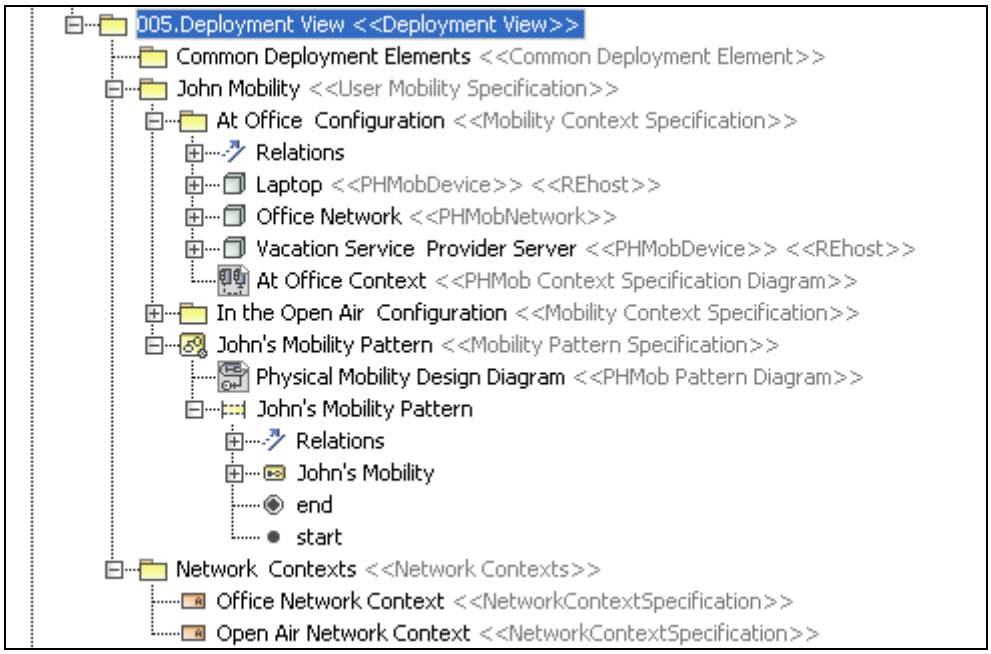
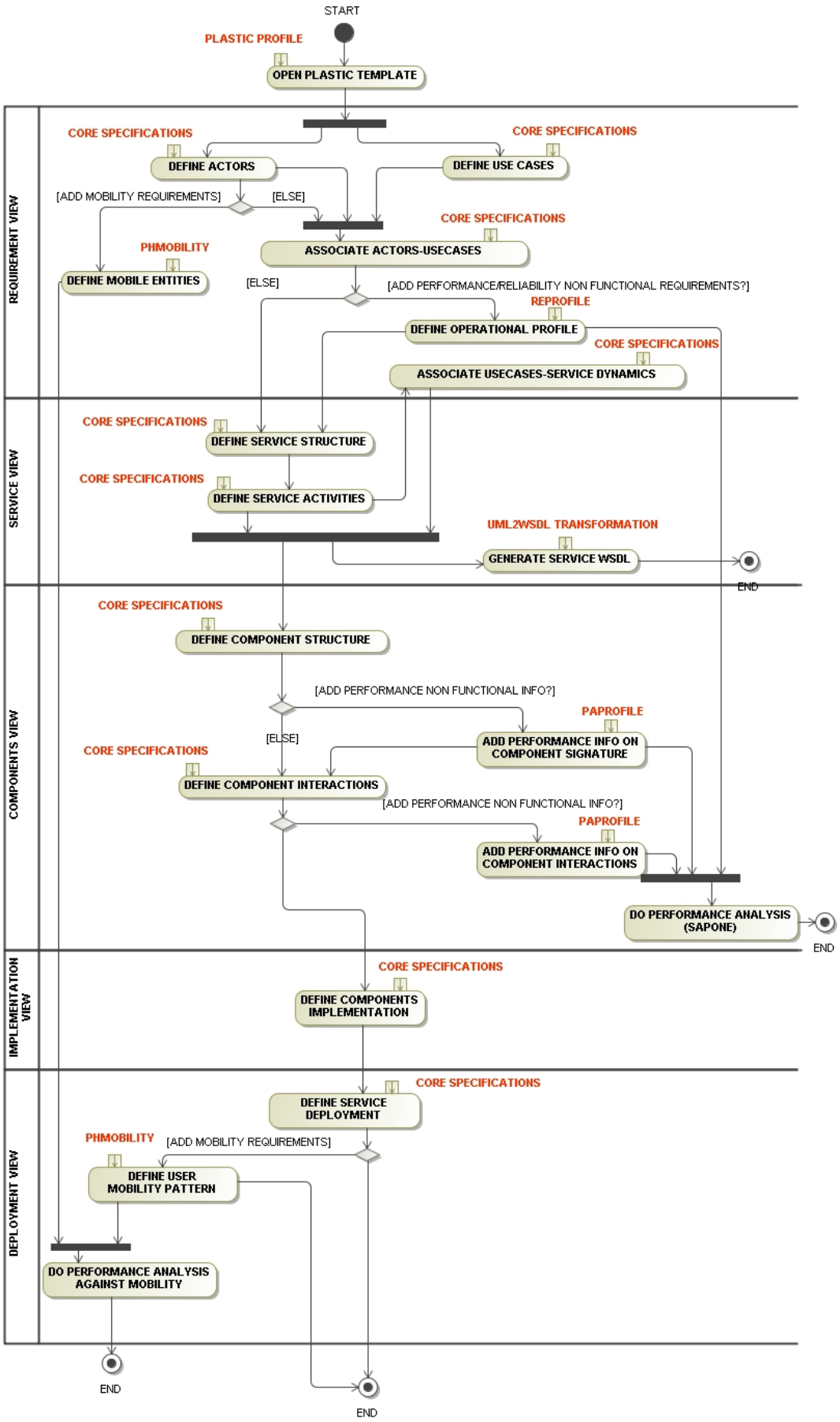


Fig 28: The VPS Deployment View


START
PLASTIC PROFILE
OPEN PLASTIC TEMPLATE
REQUIREMENT VIEW
CORE SPECIFICATIONS
DEFINE ACTORS
CORE SPECIFICATIONS
DEFINE USE CASES
[ADD MOBILITY REQUIREMENTS]
[ELSE]
CORE SPECIFICATIONS
ASSOCIATE ACTORS-USECASES
PHMOBILITY
DEFINE MOBILE ENTITIES
[ELSE]
[ADD PERFORMANCE/RELIABILITY NON FUNCTIONAL REQUIREMENTS?]
REPROFILE
DEFINE OPERATIONAL PROFILE
CORE SPECIFICATIONS
ASSOCIATE USECASES-SERVICE DYNAMICS
SERVICE VIEW
CORE SPECIFICATIONS
DEFINE SERVICE STRUCTURE
CORE SPECIFICATIONS
DEFINE SERVICE ACTIVITIES
UML2WSDL TRANSFORMATION
GENERATE SERVICE WSDL
END
COMPONENTS VIEW
CORE SPECIFICATIONS
DEFINE COMPONENT STRUCTURE
[ADD PERFORMANCE NON FUNCTIONAL INFO?]
PAPROFILE
[ELSE]
ADD PERFORMANCE INFO ON COMPONENT SIGNATURE
CORE SPECIFICATIONS
DEFINE COMPONENT INTERACTIONS
[ADD PERFORMANCE NON FUNCTIONAL INFO?]
PAPROFILE
ADD PERFORMANCE INFO ON COMPONENT INTERACTIONS
DO PERFORMANCE ANALYSIS (SAPONE)
END
IMPLEMENTATION VIEW
CORE SPECIFICATIONS
DEFINE COMPONENTS IMPLEMENTATION
DEPLOYMENT VIEW
CORE SPECIFICATIONS
DEFINE SERVICE DEPLOYMENT
PHMOBILITY
[ADD MOBILITY REQUIREMENTS]
DEFINE USER MOBILITY PATTERN
DO PERFORMANCE ANALYSIS AGAINST MOBILITY
END
END

Step 5

Once implementation view has been produced, the modeling process finish with the definition of mobility patterns of mobile PLASTIC users (PHMobEntity specified at Requirements View e.g. John) and the specification of PLASTIC enabled mobile devices as hosts that execute the Component Implementation Specifications (specified at Implementation View) deployed on it. Each PLASTIC enabled device (e.g. John's laptop or mobile) is connected to network(s) of type allowed by B3G open wireless environment: all network characteristics are modeled as network related context information (*Network Context Specification* e.g. OfficeNetworkContext). This information, retrieved by means of a suitable PLASTIC Middleware , together with its device information (specified at Component View) realize the context-awareness of PLASTIC Service being modeled and drive its adaptation.

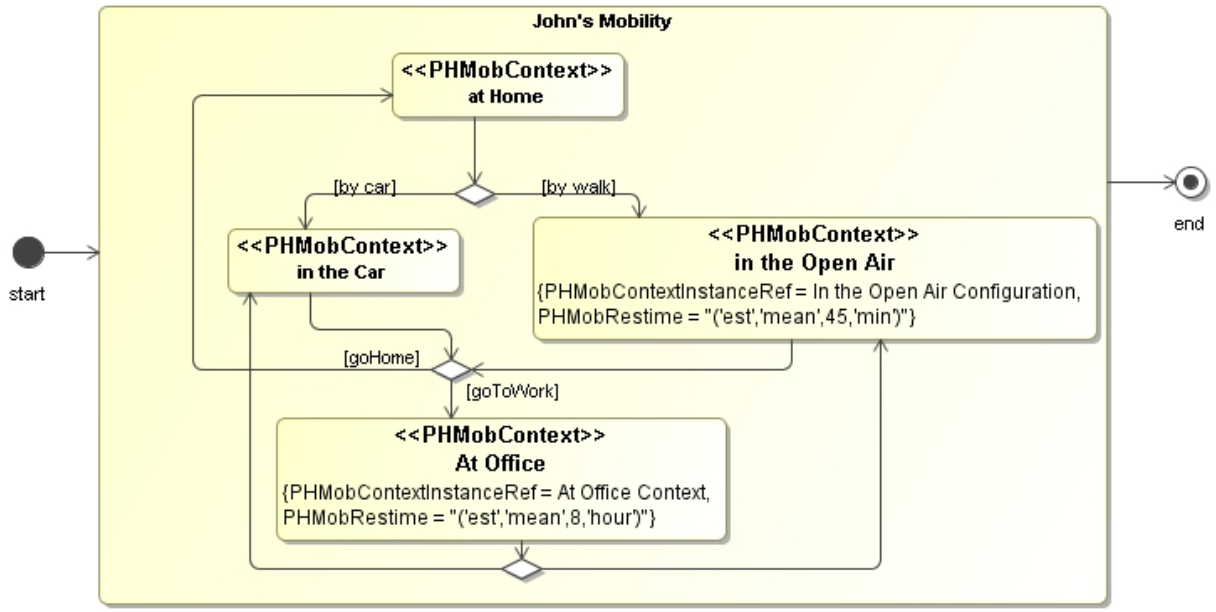


Fig 29:The Physical Mobility Pattern Diagrams of John Physical Mobility Entity.

The deployment view follows an user-centric description. While Use Case Service Diagrams (UCSD) is used at requirements view to identify service capabilities and their consumer, the information related to user physical mobility is modeled by Physical Mobility Pattern Diagrams (PhMPD) and Service Deployment Diagram (SDeployD). PhMPD describes a pattern as a set of possible transitions between different contexts (modeled by uml.State), characterized by its own hardware and software configuration (the SDeployD referenced by the PHMobContextInstanceRef tag), the user, provided with his PLASTIC enabled device, interacts with during his moves.

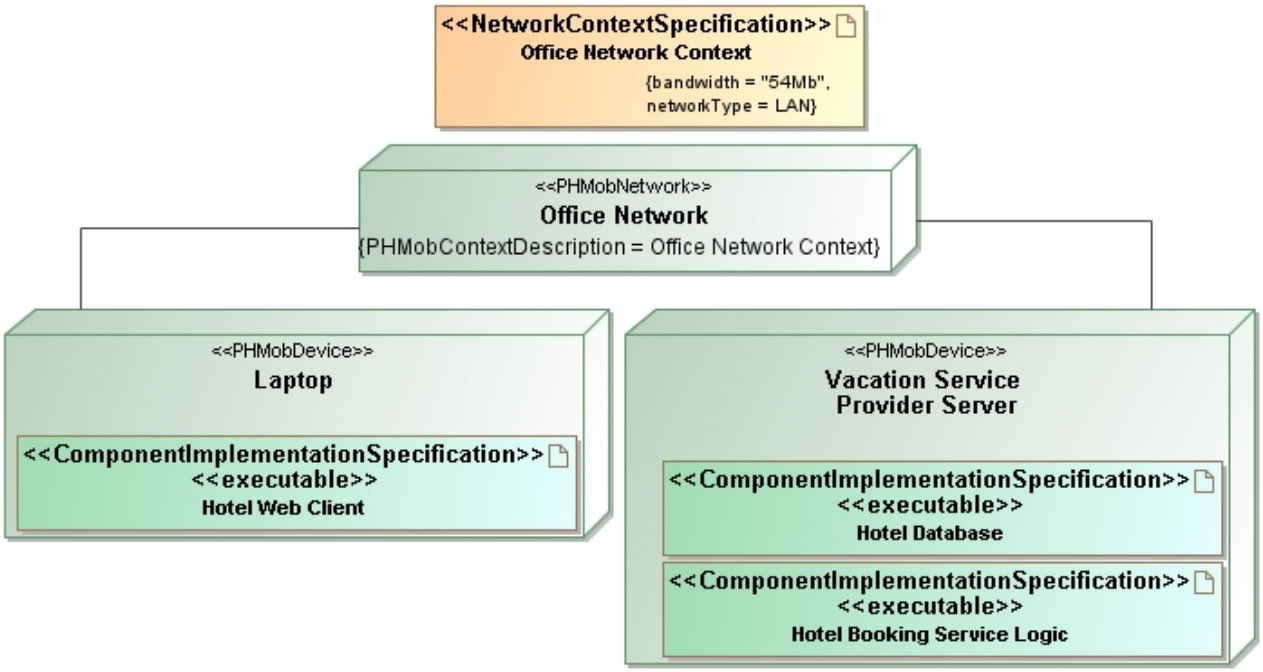


Fig 30:The Service Description Diagrams at John's Office.

After that the Physical Mobility Pattern Diagram is defined for each nomadic user (PHMobEntity), his possible contexts (PHMobContext) are detailed.
The modeling elements used to specify the user's configuration are:

(i) The *NetworkContextSpecification* (from PLASTIC Specifications, e.g. John's Office Network Context) contains all network related information that, together with device related one, defined at Component View, defines the context specification which context-aware PLASTIC Service have to adapt to.

(ii) *PHMobDevice* (from PHMobProfile, e.g. John's Laptop) adds mobility additional information to PLASTIC enabled device that consists in descriptions of its possible software and hardware configurations (PHMobContextDescription attribute).

PHMobNetwork (from PHMobProfile, e.g. John's Office Network) abstracts the networks which PLASTIC enabled devices are connected to. Different kind of networks (*NetworkType* enumeration type) can be available within the open wireless B3G environment: each different network is then described by its own NetworkContextSpecification attribute.

3.3.8.1 Model elements used at Deployment View.

In the following table, the stereotypes used at Deployment View are listed.
When not specified, the semantic is specified in [PLASTICD1.2][PLASTICD2.2].

Stereotype	Base Class	Target	Tags	Semantic
Network Context Specification	Classifier	Network Context Modeling	-	-
PHMobContext	State	Mobility Modeling	PHMobContextInstanceRef	[Mobility]
PHMobDevice	Node	Mobility Modeling	-	[Mobility]
PHMobNetwork	Node	Mobility Modeling	-	[Mobility]

Table 5 model elements used in Deployment View

4 References

1 [PLASTICDevProc] M. Autili, L. Berardinelli, V. Cortellessa, A. Di Marco, D. Di Ruscio, P. Inverardi, and M. Tivoli A Development Process for Self-Adapting Service Oriented Application. *Submitted to ICSOC '07*

2 [UML2] OMG. UML 2 Superstructure, February 2007. formal/2007-02-03

3 [SPT] OMG. UML Profile for Schedulability, Performance, and Time Specification January 2005 formal/05-01-02

4 [WS-MATE] M. Autili, V. Cortellessa, A. Di Marco, and P. Inverardi. A Conceptual Model for Adaptable Context-aware Services. In *WS-MATE*, 2006

5 [SAPONE] A. Di Marco, Model-based Performance Analysis of Software Architectures, Ph.D. Thesis, June 2005. http://www.di.univaq.it/adimarco/thesis/thesis-final-web.zip

6 [Mobility] A. Di Marco, C. Mascolo . Performance Analysis and Prediction of Physically Mobile Systems WOSP 2007

7 [REprofile]Cortellessa V., Singh H., Cukic B., Gunel E., Bharadwaj V., Early reliability assessment of UML based software models, 3rd ACM Workshop on Software and Performance, July 24-26, 2002 – Rome (Italy)

8 [Cortellessa RE] V. Cortellessa, A. Pompei "Towards a UML profile for QoS: a contribution in the reliability domain" in Proc. *4th Int. Workshop on Software and Performance* (*WOSP'04*), Jan. 2004, pp. 197-206

9 [TAWS] Test and Analysis of Web Services Baresi, Luciano; Di Nitto, Elisabetta (Eds.) 2007 Springer

10 [PLASTIC D1.2] [PlasticD1.2] PLASTIC IST STREP Project. Deliverable D1.2: *Formal description of the PLASTIC conceptual model and of its relationship with the PLASTIC platform toolset.*

11 [PLASTIC D2.1] [PlasticD2.1] PLASTIC IST STREP Project. Deliverable D1.2: *SLA language and analysis techniques for adaptable and resource-aware components.*

12 [PLASTIC D2.2] [PlasticD2.2] PLASTIC IST STREP Project. Deliverable D2.2: *Graphical design* language and tools for resource-aware adaptable components and services.